
Blockchain-based Collaborative Caching Mechanism for Information Center IoT

Yin Ying^{1,2,*}, Zhihong Zhou^{1,2} and Quanhai Zhang^{1,2}

¹*Institute of Cyber Science and Technology, Shanghai Jiao Tong University, Shanghai 200240*

²*Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai 200240*

E-mail: yyin77@sjtu.edu.cn

**Corresponding Author*

Received 31 October 2022; Accepted 16 November 2022;
Publication 14 January 2023

Abstract

The development of fifth generation mobile communication (5G) technology and Internet of Things (IoT) has enabled more mobile terminals to access the network and generate huge amounts of information content. This will make it difficult for the traditional IP-based host-to-host model to cope with the demand for massive data transmission, making network congestion an increasingly serious problem. To cope with these problems, a new network architecture, the Information-Centric Networking (ICN), a content network with content caching as one of its most core functions, has emerged. In addition, in the era when 5G and future 6G networks gradually realize the interconnection of everything, the Information-Centric Internet of Things (IC-IoT) based on ICN architecture has emerged, and a large number of IoT devices can use ICN nodes as edge devices to realize collaborative caching. The caching capacity of IC-IoT is directly related to the transmission efficiency and capacity of the whole network, and the performance of IC-IoT caching capacity is the top priority of research in this field.

To address the above issues, the research in this paper focuses on deploying blockchains in IC-IoT networks and using the consensus mechanism of

Journal of ICT Standardization, Vol. 11_1, 67–96.

doi: 10.13052/jicts2245-800X.1114

© 2023 River Publishers

custom blockchains to motivate ICN nodes and non-ICN nodes in the network to perform caching collaboratively, which in turn improves the caching capacity of the whole network.

The main work of this paper has the following points. First, incentivize IC-IoT collaborative caching based on blockchain consensus mechanism: by deploying blockchain in IC-IoT, rewarding nodes that obtain bookkeeping rights to incentivize network-wide collaborative caching, and designing experiments to compare the caching capacity of the network before and after the incentive; second, improve DPoS consensus mechanism to incentivize collaborative caching: Experiments are designed to compare the incentive capacity of PoW consensus mechanism and improved DPoS consensus mechanism for IC-IoT network collaborative caching, and to select the consensus mechanism with better performance; third, the design and implementation of IC-IoT test bed: write ICN program to form ICN network from basic communication to multiple nodes, and then deploying blockchain on the network for subsequent extension studies.

This thesis demonstrates the feasibility of using blockchain for IC-IoT network cache collaborative incentive, and proves that the blockchain incentive method in this paper can improve the throughput of IC-IoT network cache by building a test bed.

Keywords: Information center IoT, collaborative caching, blockchain, consensus mechanism, caching incentive.

1 Introduction

In today's world, Internet technology is developing at an unprecedented speed, and the rapid development of fifth-generation mobile communication technology (5G) and IoT technology has enabled more mobile terminals to access the network [1], with the consequent explosion of network traffic and network scale due to the massive information content generated by numerous terminals. However, the existing Internet architecture is still a host-to-host communication model, and the main application requirement is the sharing of computing resources, which brings more challenges to both Internet service providers and users [2]. Although the existing Internet system has adopted some Web caching technologies to enhance network robustness, these caches are often not updated in a timely manner; there is no effective mechanism to push frequently requested user content to the edge of the network, and when the cache volume reaches a certain level, the ability to obtain content is

limited by the cache bottleneck; Most important the IP addresses of servers where resources are stored are open to the public, and malicious attackers can carry out distributed denial of service attack (DDoS) to bring down servers [3]. In addition, by using the high bandwidth using 5G technology, the attack bandwidth of DDoS increases which makes the security problem faced by host-to-host model more serious [4].

In response to these problems, Information-Centric Networking (ICN) has been proposed and widely discussed [5]. ICN uses information names as the core of the transport identity and protocol structure, and focuses only on the location of the content. Any node that has cached content can immediately return the corresponding content if it can satisfy the user's request, without having to communicate with the server via IP address, and the user does not have to care about the source of the content [6]. ICN's performance advantages over traditional network architectures include high efficiency, high security, mobility support, simple deployment, and green features. More importantly, in the era of rapid development of 5G and 6G and gradual realization of Internet of Everything, the Information-Centric Internet of Things (IC-IoT) proposed based on ICN architecture was born [7]. IC-IoT not only inherits the advantages of ICN over traditional networks, but also a large number of IoT devices can be used as edge devices to assist the content caching of ICN nodes [8], which is an important link to enhance the functional system of ICN. Based on the above two advantages, IC-IoT effectively solves many of the above problems of traditional networks, and after its combination with IoT as a kind of content network, how to use a large number of IoT devices for ICN content-assisted caching is one of its most core issues [9], which is directly related to the transmission efficiency and capacity of the whole network. Therefore, this topic focuses on how to improve the collaborative caching capability of IC-IoT, enhance the availability of IC-IoT networks, effectively solve the increasingly serious network congestion problem, and improve link QoS and user experience.

Meanwhile, IC-IoT is often a distributed content storage, which has high requirements for distributed trust management and transaction support. Blockchain technology can solve the Byzantine General problem [10], used to safely store information, which can not be forged and tampered with, greatly reduces the trust cost and accounting cost of the real economy, and redefines the property rights system in the Internet era. For IC-IoT type of distributed networks, blockchain technology can well enable consensus among nodes and record the transactions between nodes. In addition, consensus mechanism as the core of blockchain technology, on the one hand, it

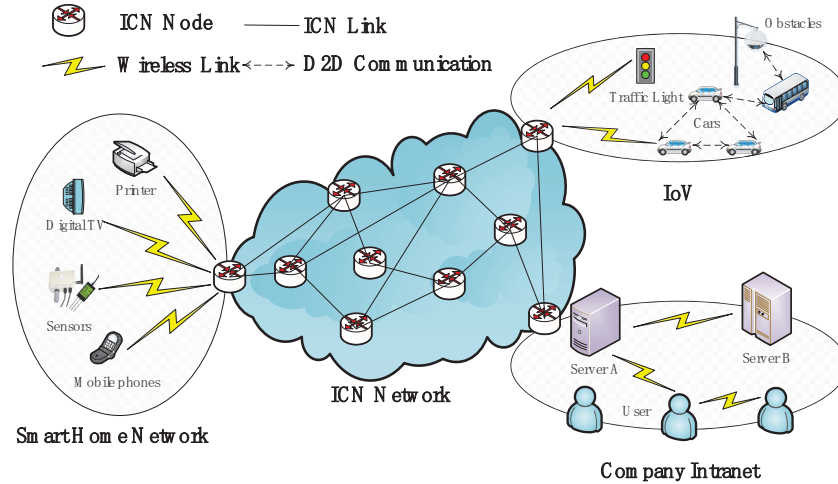


Figure 1 Schematic diagram of IC-IoT application scenarios.

can improve the security of blockchain transactions, on the other hand, the existence of rewards in some consensus mechanisms can have an incentive effect on some work.

By improving and applying the existing more mature consensus mechanism, we can design a new consensus mechanism for incentivizing the overall caching performance of IC-IoT, and reward the nodes with excellent caching ability and collaborative caching ability in IC-IoT, which can promote each node of IC-IoT to find a more effective caching mechanism and motivate ICN nodes to continuously seek surrounding IoT devices for collaborative caching, which eventually improves the caching capacity of the network. In addition, basic attributes such as the longest chain law of the blockchain itself can also ensure the security of transaction information records in the network. Therefore, promoting IC-IoT collaborative caching through the basic idea of blockchain technology can effectively enhance the overall caching utility of IC-IoT, which in turn improves user experience and utilization of edge IoT resources, and can also make its security further guaranteed.

The research on collaborative caching and blockchain applications in IC-IoT has the following main implications.

- (1) Solving the host-to-host mode problem: the continuous improvement of IC-IoT caching capacity can successfully solve the increasingly serious network congestion problem in the traditional host-to-host mode, and

can improve the processing rate of massive data in IoT in the new era of 5G and 6G.

- (2) Further improve IC-IoT caching capacity: Inspired by the application of consensus mechanism in blockchain, we improve the consensus mechanism and apply it to IC-IoT to motivate each node to cache contents, so as to cache contents more effectively and improve the success probability of users to get the requested contents and the processing speed of IC-IoT. And to increase the utilization of IoT devices in IC-IoT by motivating ICN nodes and IoT devices for collaborative caching.
- (3) Securing IC-IoT network: The introduction of blockchain not only stimulates the network to improve the caching capacity, but the decentralization feature and the longest chain principle of blockchain itself can ensure that the network content is not tampered with and the transaction information of nodes in the network is recorded safely, which guarantees the security of network operation. The successful deployment of blockchain also lays the foundation for the subsequent security scalability.

2 Research Status

2.1 Analysis of Current Research Situation Domestic and Abroad

2.1.1 Current status of IC-IoT collaborative caching research

For IC-IoT caching mechanism, many domestic and foreign scholars have made relevant researches. First, in terms of algorithm, the most primitive caching strategy is LCE (Leave copy everywhere) [11], Although this scheme is simple and easy to implement, but the redundancy makes it inefficient. In probabilistic caching (Probabilistic caching) [12] policy, each router determines the probability of caching packets based on its own caching capacity, which prevents routers with small capacity from blindly caching a large number of packets. Bilal M et al. proposed a caching scheme called least frequent recently used (Least frequent recently used), which is a combination of Least recently used (LRU) and Least frequently used (LFU), LFRU is suitable for networks with in-network caching, and ICN is one of its good application scenarios, where particularly popular content is kept in the privileged partition and not easily replaced, and ordinary popularity content is kept in the unprivileged partition and enforces general cache replacement policies. Din et al. [13] in 2018 pointed out that IoT is very suitable for application in ICN scenarios, but the traditional ICN caching strategy may

not be applicable in IC-IoT [14]. More and more scholars are delving into the study of IC-IoT caching mechanisms in order to find more efficient content caching mechanisms, reflecting the importance of research on this topic.

2.1.2 Status of research on combining IC-IoT and blockchain

There have been many examples of combining IC-IoT with blockchain to solve problems at home and abroad. In article [15], it is proposed that in IC-IoT networks, since content can be copied and cached on any node, content is no longer under the control of the content owner after it is published, so its combined with blockchain to study an IC-IoT access control scheme. The article [16] proposes peer-to-peer (P2P) overlay blockchain networks deployed on the underlying network, and with the development of P2P technology, the mismatch between the application layer and the underlying network brings a large amount of redundant communication, by combination of IC-IoT with blockchain, the mismatch problem can be solved. The article [17] proposed a blockchain-based BICN architecture, which is used to solve the security problems caused by malicious behaviors in IC-IoT. The article [18] proposes a blockchain-based framework for IC-IoT data lifecycle protection. The article [19] addresses the problem that there will be some malicious unmatched toxic contents in the existing IC-IoT network, analyzes that the traditional content inspection strategy is too costly, and proposes a blockchain-based system to remove the toxic contents from the network. In summary, the combination of IC-IoT and blockchain has been widely researched and applied.

After investigating the current state of research in these two areas, we found that the research for IC-IoT caching mechanism mainly focuses on how nodes in the network should specifically select the caching mechanism so as to improve the caching capacity of individual nodes or reduce the energy consumption of nodes, etc. The whole network is not considered from a global perspective, and the selected caching mechanisms may be different for nodes in different environments. The research on the combination of blockchain and IC-IoT is more often to use the security features of blockchain to propose solutions to some security problems of IC-IoT. There is a lack of research on stimulating IC-IoT collaborative caching capabilities from a global perspective using improved blockchain consensus mechanisms. This topic introduces the improved blockchain consensus mechanism in IC-IoT to motivate each node to find the most suitable and efficient caching mechanism, and intends to motivate IC-IoT to improve the caching capability from a global perspective.

3 IC-IoT Architecture and Collaborative Caching Strategy

3.1 Main Principles of Information Center Network

3.1.1 Overview of Information Center Network Development

Although the TCP/IP architecture has facilitated the development of computer networks, TCP/IP was not designed at the beginning to take into account the subsequent emergence of security, multicast broadcasting, IP address exhaustion and other issues, and the TCP/IP-based network architecture gradually showed its drawbacks. As shown in Figure 2, the bottlenecks of the two protocol stacks are IP protocol and Content content, respectively, in the face of different kinds of applications and link composition.

Regarding the above problems, two directions have been proposed in the related research to solve the problems, one direction is the evolutionary direction, for example, the use of IPv6 protocol [20], which expands a large number of addresses to solve the problem of IPv4 address exhaustion but with the host-to-host nature, IP addresses is still needed for communication. Another direction is the revolutionary information-centric ICN network. As shown in the diagram above, the information center network has a completely different network protocol stack than traditional TCP/IP, and IP is no longer the communication bottleneck.

3.1.2 Information center network data forwarding principle

Since the focus of this topic is on the implementation of ICN data forwarding and the incentive of ICN caching capability, this subsection focuses on the data forwarding strategy of ICN and introduces the ICN data caching method.

Differ from IP packets, there are two packet types in ICN networks, interest packets and data packets, as shown in Figure 3. When a user requests a certain content, it sends an interest packet to the network, The packet also contains a content name, which is used by the ICN router to discern which interest packets are requested and returned to the corresponding port according to the PIT table.

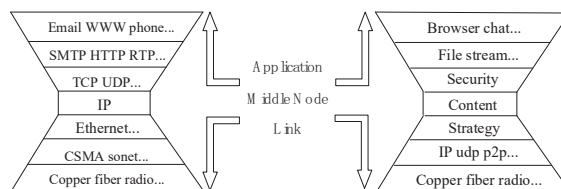


Figure 2 Comparison of IP protocol stack and ICN protocol stack.

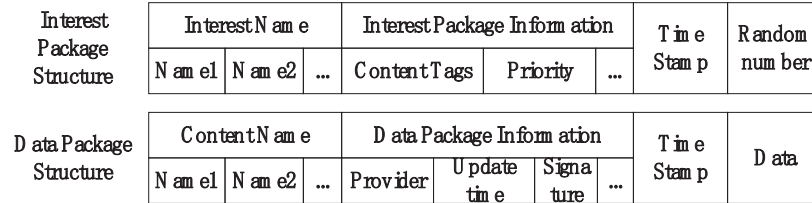


Figure 3 ICN packet of interest structure.

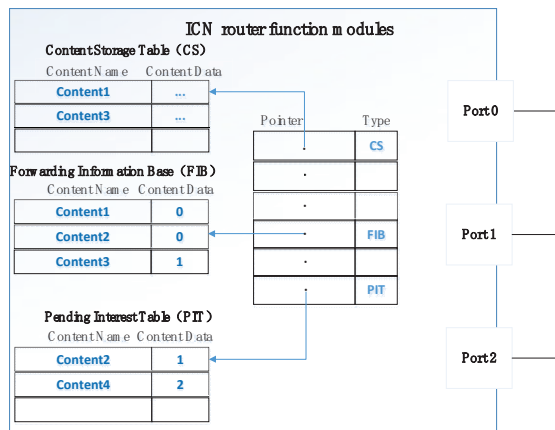


Figure 4 Sketch of ICN router internal structure.

The focus of ICN is on the content rather than the IP address, and is also mainly reflected in the data forwarding policy of ICN. Each ICN node needs to maintain three data structures: Content Store (CS), Forwarding Information Base (FIB), and Pending Interest Table (PIT).

The CS is essentially a cache, the CS in an ICN, which chooses a caching policy to determine whether the content passing through it needs to be cached locally and how to discard the cached content when the cache is full. The FIB indicates where to forward the packet of interest, compared with the FIB table of the IP router, the FIB table of the ICN router stores the prefix of the content and the content prefix is related to the naming mechanism of the ICN. After the interest packet, the forwarding record is recorded in the PIT table so that when the corresponding packet is subsequently obtained, the packet can be returned to the requester in its original path according to the request path, and the PIT will delete this request record when the packet is passed back.

The above data structure diagram is shown in Figure 4. The CS table illustrates that Content1 and Content3 are cached in this router. The PIT table

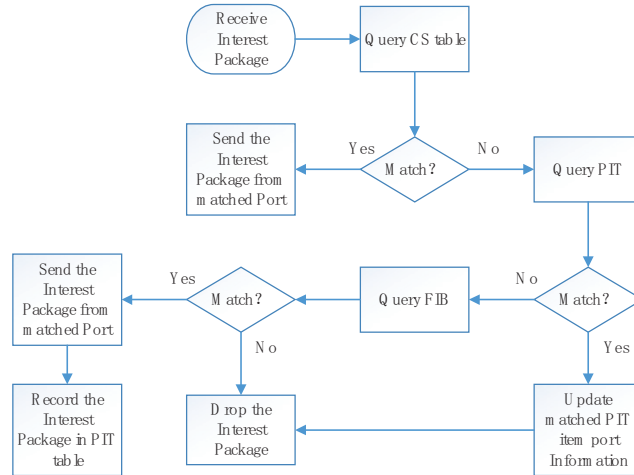


Figure 5 Flowchart of ICN router processing interest packets.

illustrates that Port 1 receives a packet of interest requesting Content2 and Port 2 receives a packet of interest requesting Content4, and when this router receives the corresponding content, it will pack the corresponding content to the corresponding port according to the record in the PIT table. The FIB table records that if there is no Content1-Content3 locally, the requested interest packets are sent to ports 0, 0 and 1 respectively.

Distinguished from IP routers that forward packets based on IP addresses, ICN router packet forwarding has the following three main steps, as is shown in Figure 5.

3.2 Information Center Network Caching Policy Study

This subsection introduces the forms of caching in ICN, analyzes two forms of collaborative caching, and describes how collaborative caching can improve the operational efficiency of the whole network.

3.2.1 LCE caching strategy(no collaborative caching)

LCE (Leave cache everywhere) is the most widely used caching strategy in ICN. During the data request process, each router caches the requested packet locally when it is received and records this content in the CS table entry. In this way, each router node in the data transmission path caches a backup copy of the content, and when a request for this content is made later, the consumer can obtain the content from the nearest router instead of having

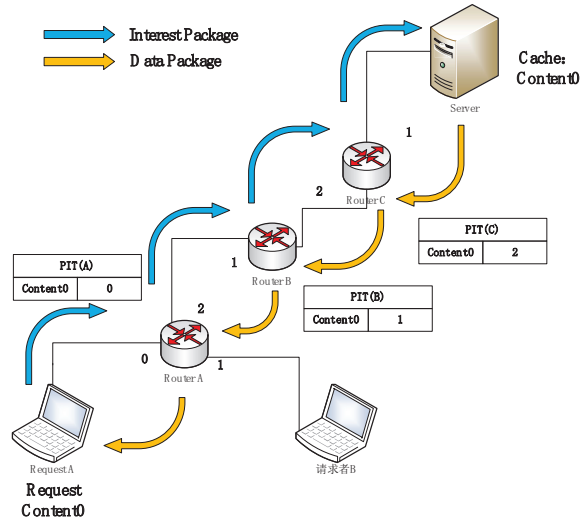


Figure 6 Diagram of ICN network content request return.

to forward it through multiple routers to obtain the content from the server again.

The LCE caching policy is simple to deploy. However, being no collaborative relationship between individual routers, each router only focuses on whether a content is cached locally, and will only decide whether to cache the content when it arrives based on whether the content is available in the local CS.

When routers A, B, and C on the path do not have Content0 cached, requester A requests the interest packet path and gets the server to return the packet path as shown in Figure 6. At this time, the LCE policy is implemented, so all three routers will cache Content0 locally, and the CS table situation of each router after the request is shown in Figure 7.

After a content request is made, all routers on the path cache the content. When another requester B makes a Content0 request, although it can quickly get the requested content from the nearest router, the content they cache cannot be requested when the rest of the routers are not connected to other requesters, as shown in Figure 7, the interest of requester B packets will only be processed by Router A. Router B and Router C are not connected to other requester nodes, and the Content0 they cache at this point tends to become a redundant cache with a very low hit rate. This results in a large amount of cache redundancy of the same content by routers throughout the path, wasting

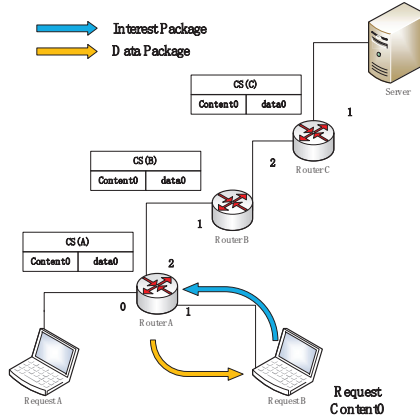


Figure 7 Network node cache situation and secondary request schematic.

a large amount of cache space and significantly decreasing cache utilization. When the number of data requests in the network increases, the restricted storage space cannot meet the demand for massive information storage and will reduce the transmission efficiency of the network.

3.2.2 Collaborative caching

According to the description in 2.2.1, some contents only need a small number of router caches in the path, and blind deployment of LCE policies will eventually bring redundancy and affect the operational efficiency of the whole network. As shown in Figure 7, the content cached in the middle path is redundant, and the content only needs to be cached at the edge of the network. At this point, the routers are divided into edge routers and core routers, with the core routers mainly responsible for interest packet packet forwarding and the edge routers mainly responsible for selecting the appropriate caching policy to cache the content. This can reduce the storage burden of the core router and improve its forwarding efficiency, on the other hand, it can improve the overall cache hit rate of the network and reduce cache redundancy, so that consumers can directly request data from the edge router to get the content within a shorter delay. Amble M. et al. [31] demonstrated that in the process of router caching content, the time required to get the content from the next-hop router is generally twice as long as the time required to get the content from the previous-hop router. router twice the time it takes to obtain the content, which suggests that the overall network caching trend is moving from core caching to edge caching.

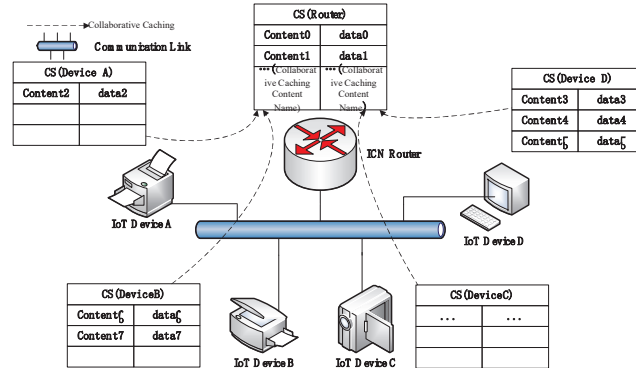


Figure 8 Schematic diagram of collaborative caching between ICN routers and IoT devices.

The focus of this paper is to motivate collaborative caching in IC-IoT, which can be divided into collaborative caching between ICN nodes and collaborative caching between ICN nodes and non-ICN nodes (IoT devices). Since most of the accessed IoT devices in IC-IoT are user-oriented application-oriented devices, they are generally at the edge part of the network, and their communication functions necessary to meet the regular functions can provide a large amount of potential cache space for the whole network, as shown in Figure 8, where the ICN router locally caches Content0 and Content1, and the rest It can be seen that for an ICN router node, collaborative caching with the connected IoT devices can directly increase its cache space, and the communication between the ICN router and the IoT devices is at the edge of the network, which does not bring additional overhead to the packet forwarding of the core router. Therefore, the focus of this research is to combine the characteristics of blockchain to stimulate the ICN router to collaboratively cache with a large number of surrounding IoT devices through an improved consensus mechanism to increase the cache space of the router and finally improve the caching capacity of IC-IoT.

4 Design of Blockchain-based IC-IoT Collaborative Caching Incentive Mechanism

4.1 Consensus-based Incentive Mechanism for IC-IoT Collaborative Caching

The incentive effect in the blockchain consensus mechanism can be used to motivate the node caching in IC-IoT. Without reward incentive mechanism

in IC-IoT, every node of ICN will only caches the content that is of interest or benefit to itself, they are unable to hit the cache for most of the time, which results in a large number of interest packets forwarded and packets transmitted, bringing more transmission pressure to the subsequent forwarding network, reducing the transmission efficiency and increasing the request latency of users. When an incentive mechanism is present to make the nodes to actively reach collaborative caching agreements with neighboring IoT devices to increase their own caching space, thus improving their caching hit rate and obtaining rewards from the mechanism.

In the case of using DPoS consensus mechanism, proxy nodes use tokens (Tokens) for collateralization, and the more tokens they collateralize, the higher their probability of becoming the next bookkeeping node, and the higher their probability of receiving rewards. Therefore, we can improve the DPoS mechanism by linking it to the cache hit rate, and stipulate that in IC-IoT, nodes that run for bookkeeping rights use the cache hit rate as collateral, and the higher the cache hit rate, the higher the probability of becoming a bookkeeping node, receiving economic rewards or cache space rewards. Nodes are bound to explore ways to improve their cache hit rate in order to obtain economic benefits, and collaborative caching with surrounding IoT devices is one of the important methods. If some nodes still adopt selfish strategies, the rest of the nodes can choose the path with shorter delay by evaluating the delay of sending and receiving packets under different forwarding paths, and eventually the selfish nodes will be isolated by the network. In summary, the improvement of the consensus mechanism is to set the collateral in DPoS as the cache hit rate in IC-IoT, and introduce the incentive mechanism to motivate the nodes in the network to improve the caching capacity. The schematic diagram 2 of the improved consensus mechanism to incentivize IC-IoT caching capacity is as follows.

4.2 Collaborative Cache Incentive Function Implementation

Firstly, the most basic blockchain is implemented using go language, and the basic block structure is as follows.

```
type Block struct {  
    Index int  
    Timestamp string  
    TX []byte
```

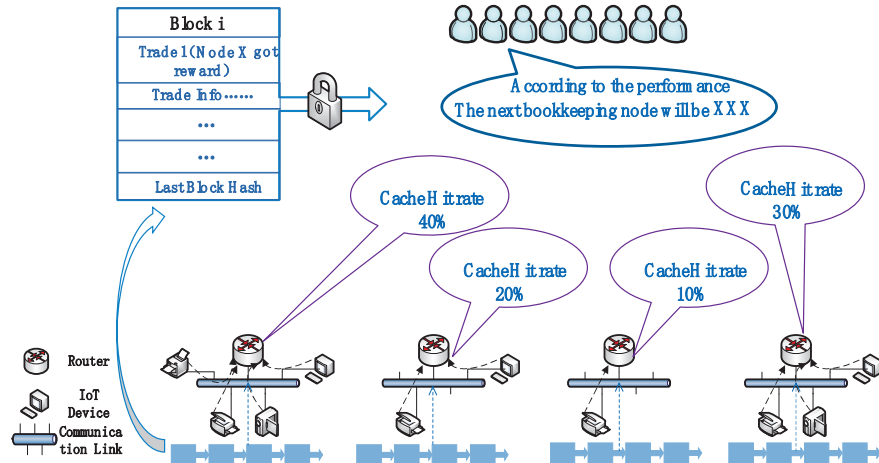


Figure 9 Schematic diagram of improved consensus mechanism to incentivize IC-IoT collaborative caching.

```

Hash string
PrevHash string

delegate *Node
}
    
```

In each block, Index is the index value, which records how many blocks in the block chain the current block is, and the index value of the Genesis block is defined as 0. Timestamp is the timestamp, which records the time when the block is generated. TX is the transaction content, which stores the relevant transaction information and consensus records in the network. hash is the hash value of the block. prevHash is the hash value of the previous block, and the PrevHash of the Genesis block is null. The PrevHash is the hash of the previous block, the PrevHash of the Genesis block is null. delegate is used to record which node in the network generated the block, the node is represented by the Node data structure.

```

type Node struct {
    Name string //node name
    CacheHitRatio int // Cache hit ratio
}
    
```

The cache hit ratio is recorded in the node, and the cache hit ratio will be used as a reference to select the nodes with good caching ability to become the network bookkeeper and achieve caching incentives.

The hash value of each block takes the content of the whole block as input and uses the SHA256 function to calculate the hash value.

```
func CaculateBlockHash(block Block) string {  
    record := string(block.Index) + block.Timestamp + block.TX + block.PrevHash  
    return CaculateHash(record)  
}
```

function as shown above, combine the block's index value, timestamp, transaction record, and the previous block's hash as a string for hash calculation.

```
func CaculateHash(s string) string {  
    h := sha256.New()  
    h.Write([]byte(s))  
    hashed := h.Sum(nil)  
    return hex.EncodeToString(hashed)  
}
```

The function to create a new block is as follows

```
func GenerateBlock(oldBlock Block, TX string) Block {  
    var newBlock Block  
  
    newBlock.Index = oldBlock.Index + 1  
    newBlock.TX = TX  
    t := time.Now()  
    newBlock.Timestamp = t.String()  
    newBlock.PrevHash = oldBlock.Hash  
    newBlock.Hash = CaculateBlockHash(newBlock)  
  
    return newBlock,  
}
```

The new block takes the old block and the new block transaction information as input, the index value of the new block is the index value of the previous block plus one, the new area fast information directly assigns the user input into it, the timestamp directly calls the time function to get it, the PrevHash directly writes the hash value of the old block, and the hash of the new block directly calls the previous hash calculation function.

In the process of block generation, it is also necessary to check the validity of the blocks to ensure that the blocks form a chain, to avoid the confusing situation of multiple blocks and multiple chains.

```
func IsBlockValid(newBlock, oldBlock Block) bool {
    if oldBlock.Index+1 != newBlock.Index {
        return false
    }
    if oldBlock.Hash != newBlock.PrevHash {
        return false
    }
    if CaculateBlockHash(newBlock) != newBlock.Hash {
        return false
    }
    return true
}
```

The function determines the block validity from three aspects, first check if the index is the previous block index plus one, then check if the PrevHash stored in the current block is the same as the hash of the previous block, and finally check if the hash of this block is correct.

The cache hit rate of the node is already stored in the Node data structure, after which it should be implemented: the higher the cache hit rate of the node, the easier it is to become the bookkeeping node in the network and get the reward.

First create a list according to the number of nodes in the network, Num, with the initial cache hit rate recorded as 0

```
func CreateNode() {
    for i := 0; i < Num; i++ {
        name := fmt.Sprintf("NODE %d | ", i+1)
```



```

NodeArr[i] = Node{ name, 0}
    }
}

```

Then, based on the number of nodes, get the cache hit rate from each node in the network and input it into the Node. See Section 4.1 for the calculation of the cache hit rate.

```

func CHRInput() {
    var CHR int
    for i := 0; i < Num; i++ {
        fmt.Scan(&CHR)
        NodeArr[i].CacheHitRatio = CHR
    }
}

```

At this point, the local list successfully obtains the cache hit ratio of each node. To simplify the whole process, we choose to sort all nodes and select the three nodes with the highest cache hit ratio to become the bookkeeping nodes to generate blocks. After that, these three nodes can create new blocks and write them to the transaction records of the whole network to obtain network rewards. Only the most extreme method of directly selecting the nodes with the highest cache hit rate is listed here. In the actual Experiment 2, a probability function needs to be set to select the bookkeeping nodes based on the specific cache hit rate data.

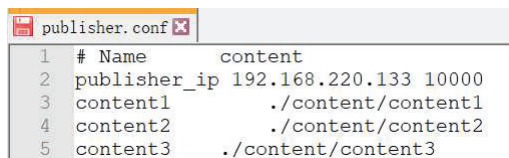
5 Experiment and Evaluation

5.1 IC-IoT Testbed Design and Principle Implementation

This subsection introduces the logic of the ICN network testbed, which is written in three parts: Publisher, Consumer, and Router. It is an overlay network based on socket programming, and each part is object-based programming, using python 2.7 language.

5.1.1 Publisher

As a content publisher, the program mainly implements two functions, one is to connect with the router, and to manage the connections from different



```

1 # Name      content
2 publisher_ip 192.168.220.133 10000
3 content1     ./content/content1
4 content2     ./content/content2
5 content3     ./content/content3

```

Figure 10 Typical data center infrastructure.

routers. The second is to retrieve the local cache directory according to the received interest packet, if there is corresponding content, send the content to the corresponding router, if not, ignore the interest packet.

The code mainly manages all sockets using a list `self.connections`. First create a main server socket `self.server_socket` to receive connection requests from the outside. After completion, add each connection request (recorded as `client_socket`) to the `self.connections` list, enter it into the `select` function, use The `select` function monitors all `client_socket` and `server_socket`.

After the read content name is stored in `data`, the location where the publisher cache content is stored is stored in the data structure `data_dic`, which is read from the pre-configuration file, as shown in Figure 10.

Since the implemented ICN network is an IP-based overlay network, the IP address of the publisher still needs to be configured in the configuration file. The latter three are example contents. In the experiment, all choose to use `contentX` (`X` is a number) to define the content.

After the content path is obtained, the content is sent to the corresponding socket through the `send` function, that is, the data packet is sent to the source of the interest packet.

5.1.2 Consumer

The consumer program needs to implement the following functions. The first is to connect with the network, and directly write the address of the router to be connected into the code; the second is to receive user input, and construct and send the request interest according to the content requested by the user. The third is to receive data packets from the router.

Connecting to the network uses the most basic socket functions. The two functions of receiving user input and receiving data packets from the router.

5.1.3 Router

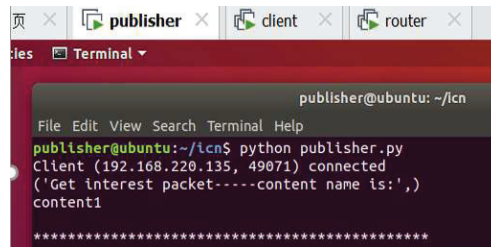
The code implementation of the router is the key to the overlay network of the entire ICN. The main functions are divided into two aspects. The first is

to maintain and monitor the related connections established with it, and to perform corresponding packet sending and receiving operations. Same as the Publisher and Consumer parts, here No longer. Another function that needs to be added is to determine whether the arriving packet is an interest packet or a data packet, and set different processing functions. The second is to realize the three tables of CS, FIB and PIT in this program.

CS, FIB, and PIT are implemented using three dictionaries. CS stores the key-value pair of content name and storage path, FIB stores the key-value pair of content name and next-hop address, and PIT stores content and connection socket. key-value pair.

The router use the select function to monitor the main socket to process new connections, and the existing sockets to process the received data. A simple demonstration is shown in Figures 11, 12, 13.

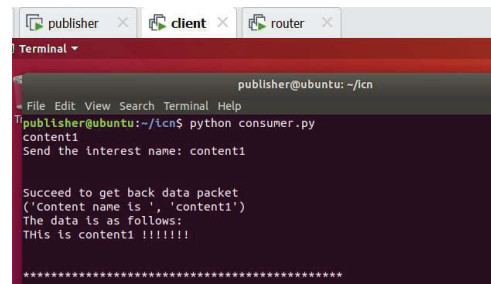
The whole ICN principle network implementation is an IP-based overlay network, so IP addresses need to be used in the connection process. In the simple demonstration, the IP of the Publisher is 192.168.220.133, the IP of the Router is 192.168.220.135, and the IP of the Consumer is 192.168.220.134.



```

publisher@ubuntu: ~/icn
File Edit View Search Terminal Help
publisher@ubuntu:~/icn$ python publisher.py
Client (192.168.220.135, 49071) connected
('Get interest packet----content name is:',)
content1
*****
    
```

Figure 11 Publisher gets Interest and responds.



```

publisher@ubuntu: ~/icn
File Edit View Search Terminal Help
publisher@ubuntu:~/icn$ python consumer.py
content1
Send the Interest name: content1

Succeed to get back data packet
('Content name is ', 'content1')
The data is as follows:
This is content1 !!!!!!!
*****
    
```

Figure 12 Consumer requests and gets content.

```

publisher@ubuntu:~/icn$ python router.py
('socket binding, host: ', '192.168.220.135', ' port:', 10000)
Client (192.168.220.134, 35449) connected

The fib table is:
{'content1': '192.168.220.133', 'content3': '192.168.220.133', 'content2': '192.168.220.133'}

The pit table is:
{}

The cs table is:
{}

('the next hop: ', '192.168.220.133')
('send the packet to ', '192.168.220.133')
*****
(<type 'exceptions.Exception'>, ', ', TypeError("cannot concatenate 'str' and '_socketobject' objects',))

The fib table is:
{'content1': '192.168.220.133', 'content3': '192.168.220.133', 'content2': '192.168.220.133'}

The pit table is:
{'content1': <socket._socketobject object at 0x77b99906b910>}

The cs table is:
{}

Succeed to get back packet
('content name is ', 'content1')
*****

```

Figure 13 Router responds to Interest and Data packets.

Figure 11 indicates that the Publisher receives a packet of interest named Content1. Figure 12 shows the result that the Consumer interface enters the content with the request name Content1 and finally gets the corresponding content successfully. Figure 13 shows the changes of the three tables in Router. The initial configuration of the FIB table instructs Router to forward the interest packets to Publisher when it receives three content requests and there is no local cache, at which time both the PIT and CS tables are empty. When the three tables are printed for the second time, there are more records of pending response Consumers in PIT. Finally, the content is retrieved from Publisher and forwarded to Consumer, and Content1 is cached in the CS table and the PIT table about Content1 is cleared.

5.2 Experimental Design

5.2.1 Collaborative caching for caching experiments

It's more efficient to let the edge routers do the main caching function in IC-IoT. When an edge router and the surrounding IoT devices generate collaborative caching, it is directly reflected in the improvement of its cache space. In our experiments we focus not on how the ICN router achieves collaborative caching with surrounding IoT devices, but on whether the rise

in cache space brought about by their achievement of collaborative caching improves the caching capacity of the network.

In our experiments, we choose the cache hit ratio as a metric to evaluate the caching capability. The cache hit ratio is defined as follows. For an edge ICN router, assuming that the content exists in its cache q times after receiving n requests, the cache hit ratio CHR (Cache Hit Ratio) is calculated as follows.

$$CHR = \frac{q}{n} \times 100\% \quad (1)$$

5.2.2 Consensus mechanism comparison experiment

The improved DPoS consensus mechanism and the cache hit ratio link can motivate each node to improve its caching capacity. In order to evaluate the effectiveness of the improved DPoS consensus mechanism, we choose to compare the improved DPoS consensus mechanism with the PoW consensus mechanism in this experiment, and record the cache hit rate of each selected bookkeeping node in the network under two different consensus mechanism deployments.

Under the PoW consensus mechanism, the main reference to decide whether a node can become a bookkeeping node is the computational power of the node (denoted as c), and the higher the computational power of the node, the higher the probability of successful mining to become a bookkeeping node.

Under the improved DPoS consensus mechanism, the main reference for deciding whether a node can become a bookkeeping node is the caching ability of the node, i.e., the cache hit rate CHR , and the higher the CHR , the higher the probability of the node being elected as a bookkeeping node. The cache hit ratio of nodes is selected using the statistics in the case of $a = 10$, and the bookkeeping node selection is performed using the cache hit ratio data in Experiment 1. The probability of node i becoming a bookkeeping node is calculated as follows (I is the node domain).

$$P(CHR - DPoS, i) = \frac{CHR_i}{\sum_{i \in I} CHR_i} \times 100\% \quad (2)$$

5.3 Experimental Results and Analysis

For the convenience of description, in Experiment I, each node is named sequentially from buffer size 10 to 20 as A-K:

After calculating the cache hit ratio of each node, the nodes A-K cache hit ratio table, sorted from expansion degree 1 to 10, is as follows.

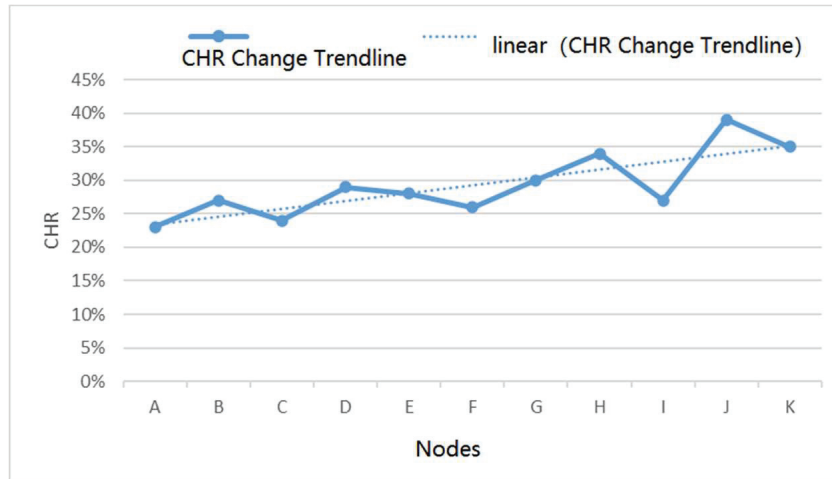
Table 1 Cache space size of each node

Node	A	B	C	D	E	F	G	H	I	J	K
Cache space size	10	11	12	13	14	15	16	17	18	19	20

Table 2 Cache hit ratio of each node

Node	A	B	C	D	E	F	G	H	I	J	K
CHR	23%	27%	24%	29%	28%	26%	30%	34%	27%	39%	35%

The line graph of cache hit ratio variation is as follows.

**Figure 14** Cache hit rate variation graph.

As can be seen from the figure, the cache hit rate rises to different degrees by enhancing the cache space of nodes through collaborative caching, and some nodes with stronger collaborative caching have lower cache hit rates, which are affected by the probability of requesting content and are within the acceptable range. The dashed line indicates an overall increasing trend of cache hit rate. This directly demonstrates the effectiveness of collaborative caching to expand the cache space to improve the cache hit rate and enhance the caching capacity.

In order to generate differences in cache hit rates among nodes, thus reflecting the effectiveness of the improved consensus mechanism to motivate caching, Experiment 2 directly follows the cache hit rate data in Table 2 in Experiment 1, as a way to participate in the bookkeeping right contention of

Table 3 Number of times each node obtains bookkeeping rights

Node	A	B	C	D	E	F	G	H	I	J	K
Number of hits Improved DPoS	10	7	6	10	6	8	11	9	10	12	11
PoW	12	6	8	5	11	13	6	9	13	8	9

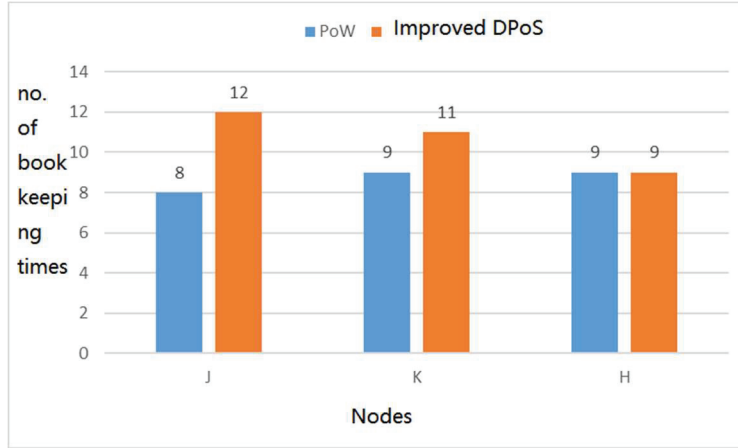


Figure 15 Comparison of the number of times the top three nodes of CHR obtained bookkeeping rights under the two mechanisms.

nodes, and the bookkeeping right contention is simulated for a total of 100 rounds under both consensus mechanisms.

The number of bookkeeping rights obtained by each node under the two consensus mechanisms is shown in the following table.

Under the improved DPoS consensus mechanism, the top three nodes in terms of the number of bookkeeping rights obtained are J (CHR = 39%), K (CHR = 35%), and G (CHR = 30%). Under the PoW consensus mechanism, the top three nodes in terms of the number of bookkeeping rights obtained are F (CHR = 26%), I (CHR = 27%), and A (CHR = 23%).

The three nodes with the highest cache hit rate are J (CHR = 39%), K (CHR = 35%), and H (CHR = 34%), and their comparative graphs of the number of bookkeeping rights obtained under the two consensus mechanisms are as follows.

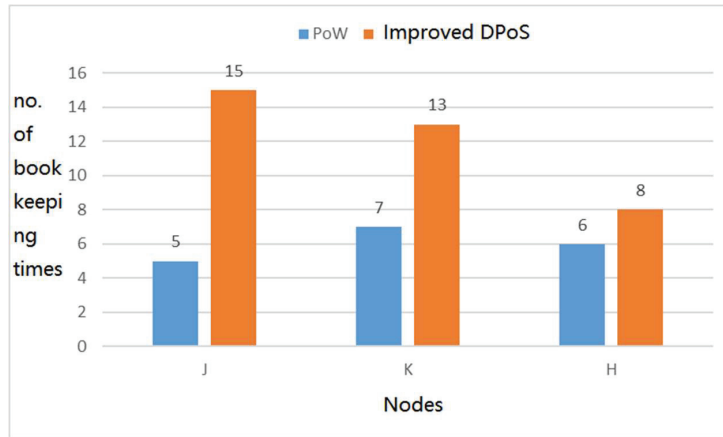
The comparison of the top three nodes and the number of nodes with bookkeeping rights under the two consensus mechanisms, shown in Figure 15. We can find that the top three nodes in the network with bookkeeping rights under the improved DPoS consensus mechanism rank in the top four

Table 4 Number of times each node obtained bookkeeping rights (second experiment)

Nodes	A	B	C	D	E	F	G	H	I	J	K
Number of hits Improved DPoS	8	12	7	13	7	9	6	8	2	15	13
PoW	16	10	8	13	7	8	9	6	11	5	7

Table 5 Number of bookkeeping rights obtained by each node (third experiment)

Nodes	A	B	C	D	E	F	G	H	I	J	K
Number of hits Improved DPoS	8	9	8	9	8	6	7	12	8	11	14
PoW	12	10	12	9	5	8	9	9	9	10	7

**Figure 16** Comparison of the number of bookkeeping rights obtained by the first three nodes of CHR under the two mechanisms (second experiment).

in terms of cache hit rate, while the top three nodes with bookkeeping rights under the PoW consensus mechanism do not have a cache hit rate of 30%. In Table 4, the number of times nodes J, K, and H with the top three cache hits were selected under both selection mechanisms are shown, and the number of times nodes J and K were selected was improved.

Due to the small difference in the cache hit rate of the ten nodes obtained in the experiment and the large chance in the election of nodes in PoW, it is easy to have the situation that the nodes with high cache hit rate are instead elected a lower number of times under the improved DPoS mechanism, and the situation that the nodes with high cache hit rate get elected more times under the PoW consensus mechanism. To ensure the credibility of the experimental results, two additional identical experiments were conducted with the following results.

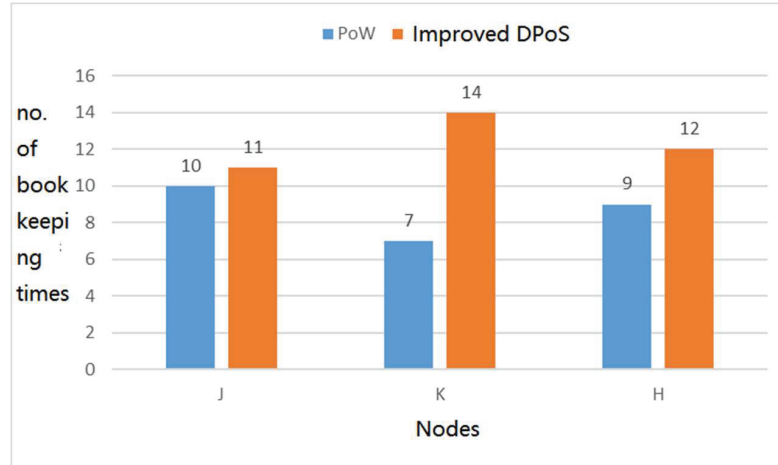


Figure 17 Comparison of the number of times the first three nodes of CHR obtain bookkeeping rights under the two mechanisms (third experiment).

In the other two experiments, it can be found that the number of times that J, K and H nodes obtain bookkeeping rights have increased to different degrees, and J and K nodes obtain bookkeeping rights steadily in the top three. Under the PoW consensus mechanism, node A with the lowest cache hit rate always gets a higher number of bookkeeping rights, and the number of bookkeeping rights acquired by nodes with high cache hit rate is full of randomness. In contrast, the experimental results illustrate that nodes with high cache hit rate are more likely to become bookkeeping nodes under the selection of the improved DPoS mechanism.

This proves that the improved DPoS consensus mechanism is more likely to select nodes with high cache hit rate as bookkeeping nodes, when the nodes in the network will try to improve their caching capacity in order to become bookkeeping nodes more easily to obtain bookkeeping rewards and fee rewards, etc.

6 Summary

In this paper, blockchain is deployed on top of IC-IoT network, and an improved consensus mechanism is used to stimulate IC-IoT network nodes to collaboratively cache, thus improving the caching capacity of the whole network, increasing the availability of IC-IoT, increasing the availability of IC-IoT, solving the problems such as network congestion that exist in the

traditional host-to-host mode, and further improving the link QoS and network user experience. An ICN overlay network and an improved blockchain consensus mechanism scheme are designed and experimentally validated for their effectiveness in stimulating the collaborative caching capability of the IC-IoT network. In the future, we can further improve the consensus mechanism and design a more efficient and simpler cache-incentivized consensus mechanism.

Acknowledgement

This project was supported by the National Natural Science Foundation of China under Grant No. U20B2048.

References

- [1] Arshad S, Azam M A, Rehmani M H, et al. Recent Advances in Information-Centric Networking based Internet of Things (ICN-IoT)[J]. arXiv, 2017.
- [2] Dizdarevic Jasenka, Carpio Francisco, Jukan Admela, et al. Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration[J]. //ACM Computing Surveys, 2019, 51(6): 1–29.
- [3] Forecast Number of 5G Subscriptions Worldwide From 2020 to 2025, 2020[EB/OL]. <https://www.statista.com/>.
- [4] You X, Pan Z, Gao X, et al. The 5G mobile communication:the development trends and its emerging key techniques[J]. Scientia Sinica(Informationis), 2014.
- [5] R. Zhu, S. Li, et al., “DRL Based Deadline-Driven Advance Reservation Allocation in EONs for Cloud-Edge Computing,” IEEE Internet of Things Journal, vol. 9, no. 21, 2022.
- [6] R. Zhu, S. Wu, et al., “Context-Aware Multi-Agent Broad Reinforcement Learning for Mixed Pedestrian-Vehicle Adaptive Traffic Light Control,” IEEE Internet of Things Journal, vol. 9, no. 20, 2022.
- [7] Mangili M, Martignon F, Paris S, et al. Bandwidth and Cache Leasing in Wireless Information Centric Networks: a Game Theoretic Study[J]. IEEE Transactions on Vehicular Technology, 2016:1–1.
- [8] Xu C, Mu W, Chen X, et al. Optimal Information Centric Caching in 5G Device-to-Device Communications[J]. IEEE transactions on mobile computing, 2018, 17(9):2114–2126.

- [9] Sharma Vishal, You Ilsun, Andersson Karl, et al. Security, Privacy and Trust for Smart Mobile-Internet of Things (M-IoT): A Survey[J]. //arXiv:1903.05362 [cs], 2020.
- [10] usenix. Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99)[J]. 1970.
- [11] Jacobson V, Smetters D K, Thornton J D, et al. Networking named content.[C]// International Conference on Emerging Networking Experiments & Technologies. ACM, 2009:117–124.
- [12] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks[C]// The second edition of the ICN workshop on Information-centric networking. ACM, 2012.
- [13] Arshad S, Azam M A, Rehmani M H, et al. Information-Centric Networking based Caching and Naming Schemes for Internet of Things: A Survey and Future Research Directions[J]. 2017.
- [14] Amadeo M, Campolo C, Quevedo J, et al. Information-centric networking for the internet of things: challenges and opportunities[J]. IEEE Network, 2016, 30(2):92–100.
- [15] Tan X, Huang C, Ji L. Access Control Scheme Based on Combination of Blockchain and XOR-Coding for ICN[C]// 2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud) and 2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, 2018.
- [16] Li R, H Asaeda. DIBN: A Decentralized Information-Centric Blockchain Network. IEEE, 2020.
- [17] H Li, Wang K, Miyazaki T, et al. Trust-Enhanced Content Delivery in Blockchain-Based Information-Centric Networking[J]. IEEE Network, 2019, 33(5):183–189.
- [18] Li R, Asaeda H. A Blockchain-Based Data Life Cycle Protection Framework for Information-Centric Networks[J]. Communications Magazine, IEEE, 2019.
- [19] Lei K, Zhang Q, Lou J, et al. Securing ICN-Based UAV Ad Hoc Networks with Blockchain[J]. IEEE Communications Magazine, 2019, 57(6):26–32.
- [20] Amble M M, Parag P, Shakkottai S, et al. Content-aware caching and traffic management in content distribution networks[C]. Infocom, IEEE. IEEE, 2012.

Biographies



Yin Ying received the bachelor's degree in Information and Electronic engineering from Shanghai Jiao Tong University in 1999, the master's degree in Information Security Engineering from Shanghai Jiao Tong University in 2014, respectively. She is currently working as an Assistant Researcher at the Institute of Cyber Science and Technology of Shanghai Jiao Tong University. Her research areas include Network Security Risk Assessment, Cryptographic Application Security Assessment etc.



Zhihong Zhou received the Ph.D degree in Electronic Electronic engineering from Zhejiang University in 2005. He is currently working as an Assistant Researcher at the Institute of Cyber Science and Technology of Shanghai Jiao Tong University. His research areas include Network Security Risk Assessment, IoT and IoV System Security Assessment etc.



Quanhai Zhang received his Ph.D. degree from the Institute of Pattern Recognition and Image Processing, Shanghai Jiaotong University in 2002. He is a Senior Engineer of the Institute of Cyber Security and Technology, Shanghai Jiaotong University. His research areas include Information Content Security Management and security assurance & services in E-government.

