Harsh Communication Environment Oriented Consortium Blockchain Construction on Edge for Internet of Things

Bo Gan 1, Qiwu Wu 2*, Xiang Li 2, Yang Zhou 1

1 College of Information Engineering, Engineering University of PAP, Xi’an, 710086, China
2 College of Equipment Management and Support, Engineering University of PAP, Xi’an, 710086, China
*Corresponding author’s e-mail: cli@hunnu.edu.cn

Abstract. In recent years, the Internet of Things has developed rapidly, and users have gradually become more precise in their control of IoT devices. The use of mobile cloud services in IoT has become a new trend. However, its information security and traceability strategies are still to be studied. This article introduces the application of edge computing architecture in harsh communication environments, and analyzes its deployment requirements for consortium blockchain in combination with IoT applications scenarios. The consortium blockchain framework for the harsh communication environment of IoT is proposed, and the overall structure, node deployment, consensus algorithm and security mechanism are introduced, the basic process is finally explained. It provides credible solutions for the information security, traceability, and sharing of IoT devices in harsh communication environments.

1. Introduction
In recent years, the Internet of Things has shown a spurt of growth, and its scope of application continues to overflow, gradually expanding from high-frequency application scenarios to harsh communication environments. For example, 5G terminal equipment has begun to be used in underground mining, and edge communications in the harsh battlefield environment represented by anti-terrorism. For harsh communication environments, the information security under communication environment on edge is very important, and the use of blockchain technology to record and trace information is the future trend.

In application scenarios with harsh communication environments, Mobile Edge Computing (MEC) [1] is able to solve problems such as poor communication, insufficient computing power, and dynamic changes in the environment. In recent years, the trend of mobile edge computing in harsh scenarios has provided favorable conditions for the deployment of blockchain.

This paper combines architecture of blockchain and mobile cloud computing to propose a consortium chain architecture under harsh communication environment of IoT, which can improve the access control security of IoT terminals on edge, store location information, communication records, and AI model parameters, also realize data sharing. Adapt to the highly dynamic and vulnerable environment at the edge, and meet the data security requirements of IoT terminals.
2. Structure of IoT edge oriented mobile cloud service
The edge network is a highly dynamic heterogeneous environment composed of both static nodes and mobile nodes. It is far away from the cloud center therefore its computing power and communication resources are limited [2]. It is an inevitable requirement for IoT applications to improve the response speed and information processing capabilities at the edge [3].

The IoT Edge Oriented Mobile Cloud Service (MCS) model is shown in Figure 1. From bottom to top, it is divided into a three-tier architecture of mobile self-organizing cloud, local micro cloud and remote cloud. The basic process is: ① When the network is in good condition, the edge node will offload the computing task to the remote cloud server through the wireless base station to realize mobile cloud computing; ② When in the DIL (disconnected, intermittent, limited) environment, use local micro-cloud nodes near the edge to perform local cloud computing or pre-processing, caching, and scheduling of mobile cloud computing; ③ When the connection with the local micro-cloud is limited or the load is too heavy, the edge node adopts self-organizing cloud to perform real-time local calculations.

![Figure 1. The overall structure of MCS model](image)

3. Harsh IoT Communication Environment Oriented Consortium Blockchain Construction on Edge
The MCS model is designed in combination with edge architecture and function. It is mainly aimed at the improvement of edge information exchange and processing capabilities in the harsh communication environment of the IoT, but does not involve information security, evidence traceability, information sharing and other mechanisms. This section focuses on the construction of the alliance chain framework for the above aspects, which fully compensates for the lack of security of MCS.

3.1. The overall structure
The proposed structure is divided into six layers: perception layer, data layer, network layer, consensus layer, contract layer and application layer.

In the overall structure: ① The perception layer is mainly responsible for the original data collection of the surrounding environment, which will be the input of subsequent model training, data storage and transmission onto the chain. It mainly includes IoT terminals such as smart wearable devices, UAVs, and smart homes. It is worth noting that the perception layer is not the only input of the system, take the access control records as an exception. ② The core of the data layer is a database that cannot be tampered with and has distributed characteristics. It assumes the function of data storage in the framework, and uses cryptographic algorithms such as asymmetric encryption and hash functions to ensure that the data information in the distributed database cannot be tampered with and be traced. ③ The network layer is mainly oriented to the network architecture of the anti-terrorism tactical edge cloud, which includes remote cloud, local micro cloud and mobile self-organizing cloud, which is essentially a P2P network. ④ The consensus layer refers to the consensus algorithm, whose purpose is to allow each node in the network to reach a distributed consistency, so as to ensure that the entire network is orderly and reliable. Two consensus algorithms, PBFT and Raft, are selected, and
they can be applied depending on the situation under different business and network conditions. ⑤ The contract layer mainly includes various scripts, smart contracts and algorithms, which can realize the programming of business logic. By setting constraint conditions, business processing can be achieved without a third-party endorsement. ⑥ The application layer includes four scenarios. The first is access control, that is, the separation of the access rights of various users to system resources; the second is data sharing, that is, the terminal performs data sharing under the local micro-cloud or mobile self-organizing cloud; the third is information storage, that is, the terminal’s communication and location information is stored for evidence; the fourth is model parameter storage, that is, the training process parameters of the AI algorithm model are stored. Different businesses are isolated by building multiple chains.

### Figure 2. The overall structure of this scheme

| Layer          | Access control | Information deposit | Model parameter deposit | Data sharing |
|----------------|----------------|---------------------|-------------------------|-------------|
| Application    |                |                     |                         |             |
| Contract       | Script, code   | Smart contract      |                         |             |
| Consensus      | PBFT           | Raft                |                         |             |
| Network        | Remote cloud   | Local microcloud    |                         |             |
|                | Mobile self-organizing cloud | Node deployment |                         |             |
| Data           | Block          | Asymmetric encryption | Digital signature     |             |
|                | Hash function  | Merkle tree         |                         |             |
| Perception     | Smart wear     | UAV                 | Smart home              |             |

#### 3.2. Node deployment

The deployment of alliance chain nodes facing the harsh communication environment of IoT mainly relies on the original "cloud-side-end" architecture of MCS. It will be explained in detail mainly from four aspects.

First, in terms of hardware deployment, the architecture is divided into three layers: remote cloud server, local micro-cloud server and IoT terminal, as shown in Figure 3-(a). The local micro cloud server and the IoT terminal are located at the edge, and the remote cloud server is located far away from the edge.

Second, as far as the public key infrastructure is concerned, due to the characteristics of consortium chain nodes that need to be admitted and the reality of user data confidentiality, it is necessary to deploy public key infrastructure in the architecture, including certificate authority (CA) and key management center (KMC), certificate revocation list (CRL), etc. Therefore, its deployment should be on the remote cloud server with the highest security. Although it would lead to a certain degree of centralization, it is beneficial to the management of the consortium chain and system security.

Thirdly, in terms of consensus node level, whether it is applying PBFT or Raft algorithm, each node such as remote cloud server, local micro-cloud server and IoT terminal is equal, and each node is displayed as a virtual pair in the mobile cloud. The nodes are responsible for achieving distributed consistency, as shown in Figure 3-(b). When the terminal is disconnected from the local micro-cloud server and is in a mobile ad hoc cloud, the mobile cloud marked in the figure becomes a mobile ad hoc cloud. At this time, in order to reduce communication occupation, new blocks of some business chains such as the information storage chain are generated. It is in a suspended state; the rest of the business
chains are running in the mobile ad hoc network, and the generation and chaining of new blocks will not be affected.

Fourth, in terms of node types, it is divided into full nodes and light nodes. Among them, the lightweight terminal is a light node. Whenever a block appears on the network, it downloads the block header and uses a distributed hash table to track the prefix node. This can greatly reduce its storage and communication pressure. Full nodes are arranged in high-performance IoT terminals, remote clouds and local micro-cloud servers, etc., to synchronize all blockchain data. In order to cope with the storage problem, non-server nodes in the full node will abandon the original data after full data verification and synchronization in some business chains, and only store the block header, such as the information storage certificate chain.

![Node deployment](image)

Figure 3. Node deployment

3.3. Consensus algorithm

In the P2P network, the role of the consensus algorithm is to make all nodes in the system reach a consensus. It faces the process of reaching consensus among distributed nodes, and the final result is a stable state of consensus [4].

The PBFT algorithm [5] can provide \((n-1)/3\) fault tolerance under the premise of ensuring liveness and safety. The Raft algorithm [6] is an easy-to-understand implementation of the Paxos algorithm. The PBFT algorithm is mainly proposed to solve the Byzantine general problem, that is, the situation where nodes may do evil. For this reason, a three-phase protocol is adopted to increase the complexity of communication in exchange for the ability to tolerate Byzantine nodes; the Raft algorithm is traditional distributed consistency that can only cope with node failures, but does not have the ability to tolerate malicious nodes. Their characteristic comparison is shown in Table 1.

Different business chains also have differences in the selection of consensus. For example, for the access control chain and the storage certificate chain, the security requirements are high, the amount of business data is small, and the communication frequency is low, which is suitable for the PBFT algorithm; for the data sharing chain, first, the large amount of data and the number of nodes decides it is suitable for algorithms with low communication complexity; the second is that the data is all general data, and its security requirements are lower than that of professional data; the third is that data requires real-time sharing, which requires high latency and TPS, so the Raft algorithm is applicable.

| Features                      | PBFT              | Raft              |
|-------------------------------|-------------------|-------------------|
| Communication complexity      | \(O(n^2)\)        | \(O(n)\)          |
| Fault tolerance type          | Byzantine node    | Failed node       |
| Maximum fault tolerance       | \(3f+1 \leq n\)   | \(2f+1 \leq n\)   |
| Initial leader node           | By number         | By response speed |
| Conditions for leader re-election | Node failure or vote | Node failure     |

3.4. Security mechanism

On the one hand, the consortium chain for IoT is under closed network, and the sensor network is
isolated from the public network. Attackers cannot access the leader node server on the public network, so they cannot use the public network computer to launch remote attacks on the master node server; on the other hand, the remote cloud server and the local micro cloud server are equipped with encrypted links, and the data is backed up and stored at multiple nodes, therefore the data cannot be tampered with. For the current common attacks on the blockchain, the architecture proposed in this study can be resisted. For example, in a Sybil attack, a small number of nodes pretend to be a large number of nodes by forging or stealing their identities to implement attack. Proposed scheme can effectively prevent sybil attacks by establishing a PKI system in a secure environment to supervise and authenticate node entry. Other safety precautions are shown in Table 2.

| Attack type          | Countermeasures                                      |
|---------------------|------------------------------------------------------|
| Sybil attack        | Establish a PKI system and strengthen access supervision |
| Smart contract attack| Verify smart contract every update                   |
| Replay Attack       | Avoid hard forks and add records such as timestamps  |
| Eclipse attack      | The node only keeps connection with the participating nodes |
| DDoS attack         | Network closure, node backup, filtering request      |

### 3.5. Basic process

This study uses multiple chains to divide different businesses. On the one hand, it can meet the complexity of the business, and on the other hand, it can increase the scalability of the framework. Figure 4 shows the flow of the four types of applications mentioned.

---

**Figure 4.** The basic process of different applications

Among them, as shown in Figure 4-(c), the location information and communication records are collected by the terminal in a fixed period $T$, recorded according to the communication situation and coordinates in the time period, and stored locally. After reaching the $T - T_0 + t$, time before the block is generated, it is broadcast in the network, each terminal node then verifies the record, and stores only
the record hash locally; the local micro-cloud server stores the original data, thereby reducing the storage pressure of terminal. If the terminal is disconnected from the local micro-cloud server at \( T - T_0 \) time, it will be postponed to the next cycle. Among them, \( T_0 \) is the estimated block put-on-chain duration parameter, \( t_a \) is the broadcast delay parameter, and \( n \) is the terminal node sequence number \((n = 0, 1, 2, \ldots )\) to avoid network congestion.

Under the condition of accessing the mobile core network, the local micro-cloud server will back up the certificate information in the remote cloud. Each terminal node only saves the certificate information of the node and the block header, and deletes the record before the \( aT \) cycle to release the storage \((a \geq 6)\). If the local micro cloud server data is damaged or lost without backup, the server will request the data again from the terminal node and restore the complete data in combination with the last backup data after connecting to the remote cloud.

4. Conclusion

Based on the MCS architecture of the mobile cloud service model for the harsh communication conditions of IoT as well as the "cloud-side-end" computing architecture, this paper comprehensively analyzes user data security requirements based on the characteristics of IoT scene and the deployment of nodes, and proposes the IoT consortium blockchain architecture under poor communication conditions. This paper takes multi-chain and multi-consensus into consideration before finally provides a trusted solution based on blockchain. It provides function of user terminal access control, data sharing, location and communication record information storage, artificial intelligence model parameter storage, etc. It can adapt to the requirements of harsh communication environment for perception, coordination, decision-making, time delay, and storage. This research is still in early stage, and the next step will be to verify and improve the architecture in applications such as smart wear and smart home.

Acknowledgments

This work is supported by the natural science basic research plan in Shanxi Province of China (No.2020JM-361), the young and middle-aged scientific research backbone projects of Engineering University of PAP (No.KYGG201905), the basic research foundation project of Engineering University of PAP (No.WJY201920, No.WJY202019, No.WJY202144), the military theory research project of engineering university of PAP (No.ILY2020085), the education and teaching fund project of Engineering University of PAP (No.WJJ202039, No.WJX202110), the research capacity improvement plan of engineering university of PAP (No.WKY202127), the PAP’s Military Scientific Research Mandatory Project (No.WJ2020A020048).

References

[1] March P, Becvar Z. (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials, 19(3): 1628-1656.
[2] Tortonesi M, Stefanelli C, Benvegnu E. (2012) Multiple-UAV coordination and communications in tactical edge networks. IEEE Communications Magazine, 50(10) 48-55.
[3] Brehmer B. (2005) The dynamic OODA loop: Amalgamating Boyd’s OODA loop and the cybernetic approach to command and control. In: Proceedings of the 10th international command and control research technology symposium. Washington. pp. 365-368.
[4] Yong Y, Xiao Chun N, Shuai Z, et al. (2018) The development status and prospects of blockchain consensus algorithms. ACTA AUTOMATICA SINICA, 44(11): 2011-2022.
[5] Castro M, Liskov B. (1999) Practical Byzantine fault tolerance. OSDI. New Orleans. pp. 173-186.
[6] Ongaro D, Ousterhout J. (2014) In search of an understandable consensus algorithm. Annual Technical Conference (14). Hollywood. pp. 305-319.