A Blockchain-Based Distributed

Information Sharing System

Haiyang Sun

([hsun27@syr.edu](mailto:hsun27@syr.edu))

**Abstract**

Nowadays information sharing become more and more important, especially in domains like medical and health care. While currently, the mainstream architecture of information sharing systems are centralized. Comparing with the distributed system, centralized systems are more likely to suffer from cyber-attack and cause the “single-point-of-failure” [1, 2] problem. But it is still difficult to share information through a distributed system because of a series of problems. In this work, we will discuss a new Blockchain based information system architecture and try to solve these problems.

**1 Problems**

Currently, there are three main problems that hinder the development of distributed information systems

The first one is, a large amount of information sharing systems now use centralized architecture. Indeed, centralized architecture has its advantages like easy to set up, fast speed and the support of complex data structure. But it also has an essential shortcoming. Centralized systems are more likely to suffer from cyber-attack, once the “center” of the system be attacked and controlled by attacker, it will cause a disastrous consequence. For now, since the “center” of system usually provided by a trusted third party (like VISA provided a trusted transaction system for credit cards). Many institutions or users choose to trust their security. But it is undeniable that the risk is still exists.

Another one is the complexity of data structure. For example, the health care information sharing system, each institution is actually a “node” in this distributed network. They may want to share different type of information. That means the structure of data may verify. Since there are strict rules to limit the data structure of information in distributed information sharing system. It is hard to promote distributed architecture in many institutions.

The last problem is, many institutions are reluctant to share their information[3]. In other word, the permission mechanisms in current distributed architecture is not good enough. Some institutions may want to share just a part of their information but not all of it.

**2 Goals**

The goal of this work is to use a blockchain based platform to develop an information sharing system. The blockchain’s distributed character makes the data safe and immutable. Every node can join and leave the information sharing network freely. They can also publish data and access data from other nodes if get the permission.

**3 Proposal**

**3.1 Background**

**3.1.1 Blockchain**

Satoshi Nakamoto’s paper [4] introduced a peer to peer electronic cash system, we call it “bitcoin”. It is a cash system based on blockchain. Blockchain is essentially a distributed transaction ledger. In this structure, every block contains not only a series of transaction metadata but also the hash of its previous block. That makes blockchain immutable. Because change data in one block will cause all hash data to change in the block behind it. So, if attacker only modify one block, the system will detect it immediately and make this block invalid. Only one way to make the change valid is keep change the hash of the block behind it. It is nearly an impossible thing because all nodes in the blockchain system are competing to generate the next block, the proof of work and proof of activity [5] mechanism choose the longer chain to trust. That is, as long as the attacker didn’t control 51% of calculate power in this system, we can sure that the data is immutable.

We still can not trust the blockchain system if it is set on a small scale of network. Because attackers may control more than 51% calculate resources in this system. But for a large scale network, for example, the bitcoin network, we can say that the system is safe.

For a blockchain based distributed information sharing system, we don’t need the money-based transaction, just need to store data in block’s metadata. Because each transaction is trackable, and every node can access other node’s data if get the permission. In this case, blockchain is a good container to store and share data.

**3.1.2 MultiChain**

MultiChain [6] is an open platform for building blockchains. Users can set up private blockchain on their own computer. In MultiChain’s official website, it provides a series of command line interfaces (aka. cli) to help users to set up a private blockchain network very easily.

Because MultiChain is a blockchain based platform, data stored in MultiChain is safe and immutable, the distributed architecture protect the data from “single-point-of-failure” or “single-point-of-control” problem. Also, MultiChain provide a special data structure “stream” that can store key-value structured data. In this data structure, key can be any string and value must be hexadecimal numbers. As we discussed before, one problem to hinder distributed architecture to develop is the complexity of data structure, but in MultiChain, users can assemble their information into hexadecimal numbers and store it in stream. MultiChain also has a good permission control mechanism, it optionally controls who can connect, send and receive transactions, create assets, streams and blocks, each blockchain is as open or as closed as the user need. In real implementation, it satisfies the requirement of distributed information sharing system.

**3.2 Implementation**

**3.2.1 MultiChain-Explorer**

Once we set up own private blockchain with MultiChain, it’s inconvenience for users to track transactions only by command line interfaces. Beside, MultiChain’s command line interface didn’t work very well on every mainstream operation system [7]. In this case, we need MultiChain-Explorer [8]. MultiChain-Explorer is a web explorer that allows user to track each of transactions in MultiChain through a user friendly graphic interface.

**3.2.2 MultiChainJavaAPI**

Because MultiChain itself only support command line interfaces, users can only pass commands through command line. But for software development, it is not enough. Software engineers want an API that can be called in programming language and use programming language to call MultiChain’s commands. Here we have MultiChainJavaAPI [9] which defines almost every command in MultiChain. Through import the jar packages of the MultiChainJavaAPI, developers can call functions and pass augments to MultiChain, that makes MultiChain more effective to use.

But MultiChainJavaAPI also has shortcomings, in some functions, it tries to pass commands to command line, but some initial augment’s value is locked in its source code. For example, the “liststreamkeyitems” command. In command line API, if put a “\*” as input, it will automatically list all items with the given key, no matter how many items it has. But in MultiChainJavaAPI’s source code, this value was written as 10, so each time the liststreamkeyitems function were called, if user does not give the augment, the program will only take back at most 10 items.

**3.2.3 Store/retrieve strategy**

Since MultiChain provide a good container to store data, and its relevant software like MultiChain-Explorer help us to visualize the chain. We need a well performed store/retrieve strategy to make the data’s store and access more efficiently. The simple way is to let all the node store their data (because data is stored in metadata part of a transaction, so store data is publishing a transaction to MultiChain, so store data also called “publish”) to a single stream. This method works well if the amount of data is small. But when data or number of nodes increase, the time cost will increase significantly.

The reason of time increase is, when a new node connects to the MultiChain network, if it wants to access a specific stream and get data from it. It needs to ask the host (the node that create this stream) to subscribe this stream. when node B subscribe a stream S that is created by node A, node B will get a completely copy of stream S, for example, stream S’ and store the stream in node B’s own database. furthermore, if other nodes (include node A) publish their data to stream S, in node B’s database, stream S’ will automatically refresh and add the new data. So, when nodes are publishing data to a single stream, every node will do publish process and subscribe process simultaneously. Sometimes will cause the program to be blocked.

An improved method is each node create their own stream and do the publish process first. Then if need data from other streams, send a request and ask for subscribe. This strategy’s main idea is to separate publish and subscribe. To make block’s possibility lower.

Since MultiChain only provide search by key, that is, given the stream’s name and the key’s name, MultiChain can find the value very quickly. But it is hard to search by value. In stream, value is transferred into hexadecimal number. To implement multi-property’s query. One strategy is to store the key strings in each searching epoch, and in the next epoch, use these keys to query for the next property.

**4 Summry**

In this project, we developed a blockchain-based information sharing system. It uses MultiChain as the platform and support distributed data store and retrieve. In conclusion, blockchain provide a relatively secure architecture for data sharing. But it still has several disadvantages. There are three main shortcomings for this system:

* Comparing with centralized system, the store/access speed is slower.
* Currently, MultiChain didn’t provide delete or modify for transaction.
* Each node needs to have a complete copy of the whole chain, at least the copy of whole streams. Comparing with centralized system, it needs more memory space.

Although blockchain based information sharing system has these disadvantages, it can’t be deny that this architecture is a new way for sharing data. We believe that blockchain can play a fundamental role in information sharing systems.

**5 Future Directions**

There are still many new method to make blockchain based information sharing system better. For the three main shortcomings above, we do have ways to improve them. Design a better store/retrieve data algorithm will make the process faster, although it still cannot catch up the speed of centralized architecture, we can make it to an acceptable speed. For the requirement of extra memory space problem. Bitcoin now have a SPV (Simplified Payment Verification) protocol [10]. Each node need only store the data relevant to itself rather than the entire chain data. Literally this idea can be implemented in MultiChain and save some memory space for each node.

**References**

[1] Fromknecht C, Velicanu D, Yakoubov S. A Decentralized Public Key Infrastructure with Identity Retention. IACR Cryptology ePrint Archive 2014;2014:803.

[2] Luu L, Narayanan V, Baweja K, Zheng C, Gilbert S, Saxena P. SCP: a computationally-scalable Byzantine consensus protocol for blockchains. Cryptology ePrint Archive, Report 2015/1168; 2015.

[3] Chris Clifton, Murat Kantarcioˇglu, AnHai Doan, Gunther Schadow, Jaideep Vaidya, Ahmed Elma- garmid, and Dan Suciu. Privacy-preserving data integration and sharing. In Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pages 19–26. ACM, 2004.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[5] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin’s proof of work via proof of stake. ACM SIGMETRICS Performance Evaluation Review, 42(3):34–37, 2014.

[6] Dr Gideon Greenspan, Founder and CEO, Coin Sciences Ltd. MultiChain Private Blockchain — White Paper;

[7] MultiChain: Open platform for building blockchains. Retrieved from <https://www.multichain.com/developers/>

[8] MultiChain Explorer: <https://github.com/MultiChain/multichain-explorer>

[9] MultiChainJavaAPI: <https://github.com/SimplyUb/MultiChainJavaAPI>

[10] SPV, Simplified Payment Verification: <https://bitcoin.org/en/glossary/simplified-payment-verification>