Dewblock: A Blockchain System Based on Dew Computing

Yingwei Wang
School of Mathematical and Computational Sciences
University of Prince Edward Island
Charlottetown, Canada
Email: ywang@upei.ca

***Abstract*—The blockchain technology enabled cryptocurrencies and a lot of other applications that trust is needed among different entities. Because every blockchain client needs to keep huge amount of blockchain data, some personal computers and mobile devices cannot be used to run blockchain clients. To make things worse, the size of blockchain data is always increasing. In this paper, a new kind of blockchain system, Dewblock, is introduced. In this system, a blockchain client does not need to keep the blockchain data and it also has the features of a blockchain full node. Dewblock was developed based on dew computing principles and architecture.**

***Index Terms*—Blockchain; Dew computing; Cloud-dew architecture; Cloud services; Blockchain full client; Blockchain lightweight client.**

I. Introduction

Blockchain was first introduced with Bitcoin, a cryptocurrency, but blockchain is not limited to cryptocurrencies. It can be used in various occasions. Ginni Rometty, the CEO of IBM, once said: "Blockchain will do for transactions what the internet did for information" [1].

Blockchain has a feature that limits the range of its applications: each blockchain full client has to keep the whole blockchain starting from the genesis block; this blockchain gets longer and longer with the operation of the blockchain network. For this reason, a blockchain full client is not suitable to be deployed to personal computers and mobile devices.

It is desirable to find solutions to tackle the above problem so that the data size of a blockchain client can be reduced. Such kind of solutions are hard to find because this problem comes with the essential nature of blockchain. Blockchain data itself is the heart of the blockchain technology; the data size of a blockchain client is inherently big and inherently keeps increasing.

For some cryptocurrency systems, such as Bitcoin and Ethereum, their blockchains already have huge amount of data; a quite powerful computer is needed to run a full client. To make these cryptocurrency systems accessible by users, blockchain lightweight clients were developed and are widely used. These lightweight clients do not need to keep the whole blockchain data so that they can be deployed to personal computers and mobile devices. These lightweight clients include SPV (Simple Payment Verification) wallets, such as Electrum, Copay, and so on.

All the lightweight clients are not qualified as full clients. They do whatever the majority of mining power says. They rely on the support provided by full clients. These lightweight clients are necessary and they are playing important roles in the cryptocurrency systems. But the goal of this paper is not to find another lightweight client.

Blockchains can be used in wide range of areas. Various blockchain systems will be developed in the future for different kinds of transactions. We want to propose a generic blockchain client architecture so that these clients can be deployed to personal computers and mobile devices and these clients still have features of full nodes.

With such goals in mind, we would like to introduce a new blockchain system: Dewblock [2]. Dewblock's dew clients do not keep blockchain data so that their data size is very small; Dewblock's dew clients still have features of full nodes; Dewblock is based on dew computing principles and architecture [3][4].

The rest of the paper is organized as follows: Section II discusses the two models of blockchains and indicates that the Dewblock approach can only be applied to blockchains based on one model. The good news is that the model of a blockchain can be changed. Section III, Section IV, and Section V introduce the key concepts of Dewblock. Section VI introduces the Dewblock project and its resources. Finally, Section VII is devoted to conclusions.

II. State-keeping Models

The approach to control client data size that we are going to introduce in this paper cannot be applied to arbitrary kinds of blockchain systems. To determine each blockchain's suitability to our new approach, we discuss the state-keeping models of blockchains in this section.

Blockchains can be considered as state transition systems or state machines [5]. Different state-keeping models can be used. Two types of state-keeping models are popular in todays blockchain networks. The first model is the *unspent transaction output (UTXO) model*. The second one is the *account model*. For example, Bitcoin uses the UTXO model [6], and Ethereum uses the account model [5].

Being the first blockchain system, Bitcoin is operated using the UTXO model. In the UTXO model, each transaction spends output from prior transactions and generates new

outputs that can be spent by transactions in the future. All of the unspent transactions are kept in each full client. A user's wallet keeps track of a list of unspent transactions associated with all addresses owned by the user, and the balance of the wallet is calculated as the sum of those unspent transactions.

The account model, on the other hand, keeps track of the balance of each account as a global state. When a transaction is being verified, the balance of an account is checked to make sure it is larger than or equal to the spending amount of the transaction.

Each of these two models has its advantages and disadvantages. The features of these two models have been discussed in literature [7][8]. From our viewpoint, we believe that the difference between these two models is that they have different thinking logic or different philosophy: the UTXO model is history oriented; the account model is reality oriented.

As Satoshi Nakamoto mentioned in his historic paper [6]: "We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership." Using an analogy, if you want to verify a coin is true in our daily life, you have to go over all the transactions this coin went through: the transaction that person A gave the coin to you; the transaction that person B gave the coin to person A, and so on, until the coin was made in the mint.

This logic works, and might be meaningful in some sense, but it is in contrary to our daily practice. Using this model, all the transactions since the start of the blockchain should exist and ready for verification. Banks usually keep detailed transaction history for quite a long time, but no bank will keep all its transactions forever. As time goes by, the size of the transaction history gets bigger and bigger. This model is not sustainable; at least it is not sustainable for most clients in a blockchain network.

Using account is our daily practice in keeping records. People's money saved in bank accounts. Each student has a profile in his/her university and this profile is called an account. The key points of an account-based system is that the accounts reflect current state of the system and these accounts do not rely on the complete history of the system, whether the system is a bank, an organization, or a blockchain.

Our efforts to develop small-data-size blockchain clients shall be based on the account-model blockchains instead of the UTXO-model ones. This restriction does not limit this approach's application because UTXO-model blockchains can be converted into account-model blockchains.

The new blockchain system we are going to introduce, Dewblock, was developed from another blockchain system: Naivecoin [9]. Naivecoin uses UTXO model. To reach our goals, we have switched Naivecoin from the UTXO model to the account model. For convenience of discussion, we would like to give a name to the Naivecoin system that has been switched to account model: *Account-Naivecoin*.

The successful conversion of Naivecoin from the UTXO model to the account model shows that the new approach we have introduced in Dewblock can be applied to all blockchain systems, although a conversion might be needed. In the future, when new blockchain systems are designed for various kinds of transactions, account model shall be used if we want to adopt the Dewblock approach.

III. Dewblock Architecture: Cloud-dew Architecture

Dewblock is designed based on cloud-dew architecture [3]. A Dewblock package is composed of a cloud server and a dew server; the cloud server and the dew server talk to each other through a new type of message channel; the dew server operates in two different modes. These topics will be discussed in the following subsections.

A. Cloud Server and Dew Server

To introduce cloud-dew architecture to Account-Naivecoin blockchain system, we would use two copies of Account-Naivecoin client. Using the terminology of cloud-dew architecture, we call one Account-Naivecoin client *cloud server*, and call the other Account-Naivecoin client *dew server*. Here we need to clarify a few terms. The term *client* which appeared in blockchain client, full client, lightweight client, Account-Naivecoin client means a client of the blockchain network. The term *server* which appeared in cloud server and dew server means a server to a user. Thus a cloud server or a dew server could act as a blockchain client; a blockchain client could be considered as a cloud server or a dew server. In the rest of the paper, terms cloud server and dew server refer to software or program package; terms dew client and full client refer to this software's role in a blockchain network.

The names cloud server and dew server make sense because the cloud server usually be deployed to a public cloud service, such as Amazon Web Services or Google App Engine, or a private cloud service where the environment is configured to support such cloud servers, and the dew server usually be deployed to a personal computer or a mobile device.

The dew server is basically a copy of the cloud server but is not necessarily the exact copy of the cloud server. In this case, we would like to introduce an important and interesting difference between the cloud server and the dew server: The cloud server contains the blockchain but the dew server does not contain the blockchain. In such a system, the data size of the dew server would be quite small.

Before we go further from here, we should make one thing clear: Account-Naivecoin uses account model, but it does keep the blockchain. Can a dew server operate without a blockchain? In other words, is it possible for an Account-Naivecoin client without the blockchain to work with the rest of the Account-Naivecoin blockchain network?

The answers to the above questions are both positive. We may modify the dew server copy of the Account-Naivecoin so that it does not keep the blockchain but otherwise it still operates in the same way. In an Account-Naivecoin blockchain

network, if one client throws out the blockchain, it can still operate well in the network: it can make transactions; it can maintain the transaction pool; it can communicate with other clients; it can verify if a transaction is valid; it can even mine a new block. The only problem this client has is that when another client asks this client to provide the whole blockchain, this client cannot respond properly.

An Account-Naivecoin client without the blockchain can operate for most of the cryptocurrency functions, but it is not a blockchain full node. This client does not have enough strength to fight attacks. If only a few clients work this way, the whole blockchain network would still run properly; if many clients are not full nodes, the whole blockchain network would deteriorate, and the trust brought in through blockchain would be gone.

In Dewblock, a dew server is not a blockchain full node, but a cloud server is. The pair of a cloud server and a dew server can also serve as one single full node to the rest of the Dewblock network. In blockchain terminology, node and client were considered the same. In Dewblock, node and client are not always the same any more. A *blockchain client* is a program that a user installed in his/her computer or device to operate a blockchain network. A *blockchain node* is a logical unit that acts as one single identity in a blockchain network. A node may contain a cloud server and a dew server, but a client is always a dew server.

B. Message Channels

In an Account-Naivecoin network, only one kind of message channel exists: *inter-node channel*. When the cloud-dew architecture was introduced, another kind of communication channel was needed for cloud servers and dew servers to collaborate. The new kind of message channel between cloud servers and dew servers is called *cloud-dew channel*. WebSocket protocol was used to create such channels.

In an Account-Naivecoin network, five kinds of messages travel through the inter-node channels. In a Dewblock network, four more kinds of messages were added and they could travel through the inter-node channels and the new cloud-dew channels. Some messages can travel in both kinds of channels; some messages can only travel in one specific kind of channel. The details of the message mechanism can be found in the website http://www.dewblock.com. In the rest of this section, Section IV, and Section V we will explain the rationals related to the four kinds of newly-added message types.

Two new message types are introduced to transfer account information through cloud-dew channels:

- QUERY_ACCOUNTS
- RESPONSE_ACCOUNTS.

To make sure that the account in a dew server is consistent with the account in a cloud server, the dew server will periodically send its account information to the cloud server for verification. If discrepancy is found, the account information in the cloud server will be fetched to the dew server to replace the account information in the dew server.

C. Simple Node and Cloud-dew Node

In some situations, a dew server may want to operate as a full client for various reasons. Such option should be provided to users in case it is necessary. Thus a dew server can operate in one of the two modes: *dew mode* and *full mode*.

When a dew server is in dew mode, it behaves as described in Section III-A and Section III-B, and we call this dew server a *dew client*. The cloud server and the dew client constitute a single Dewblock node, and we call this node a *cloud-dew node*.

When a dew server is in full mode, it behaves the same with an full Account-Naivecoin client, and we call the dew server a *full client*. In this mode, the dew server itself constitutes a Dewblock full node, and we call this node a *simple node*. The cloud server is not involved in the operation of a simple node; it may be turned off, may be operated as a separate node, or may be even not deployed at all.

From here on, we may use terms *dew client* or *full client* to replace the term *dew server* whenever it is appropriate. With these terms, the mode of the dew server is indicated.

A dew server can change its mode: it can be switched from full mode to dew mode, or vice verse.

When a dew server is switched from local node to dew mode, it needs to establish the cloud-dew channel with the cloud server described in its configuration file, to establish its connections with other nodes as described in Section IV, and to discard the blockchain to become a small-data-size dew client.

When a dew server is switched from dew mode to full mode, it needs to obtain the blockchain from another place. One of the possibilities is to establish an inter-node channel with the cloud server to obtain the blockchain. The cloud-dew channel between the cloud server and the dew server shall be cut off. It also needs to re-establish its connections with other nodes according to its new role.

IV. Pair Connection Protocol

There are two types of nodes in a Dewblock network: simple nodes and cloud-dew nodes. Here we discuss the process to establish connections between nodes. First, proper connections can be described in the following:

- When two simple nodes are getting connected, one inter-node channel is needed to connect them.
- When one simple node and one cloud-dew node are getting connected, two inter-node channels are needed: one to connect the simple node to the dew client of the other node and one to connect the simple node to the cloud server of the other node.
- When two cloud-dew nodes are getting connected, two inter-node channels are needed: one to connect the two dew clients of the two nodes and one to connect the two cloud servers of the two nodes.

We make a few assumptions:

- Connections are initiated by dew clients or full clients.

- A client knows its own mode. If it is in dew mode, it knows the address of its cloud server through its configuration file.
- A client needs to know the address of another client to establish a connection.
- A client does not have to know the types of other clients (full clients or dew clients) , although it may get to know their types after exchanging messages.

We need to create rules so that connections between nodes can be properly established. For the convenience of discussion, we use an analogy to describe the above situation.

In a community, people need to get connected. We make the following assumptions:

- Every person is in a family. A family could have a gentleman and a lady or a single lady. A family cannot only have a single gentleman.
- Every family has a contact person. A gentleman or a single lady is the contact person. Connections could be initiated by any contact person to any other contact person. Somehow contact persons can find each other. Each contact person decides if a connection request will be accepted.
- A proper connection between two families can be described in the following: a gentleman is connected to a gentleman; a lady is connected to a lady; a gentleman is connected to a lady only when the lady is single.

We create the following rules so that proper connections among families can be established.

Gentleman's Rule:

- Whenever he is involved in a connection, actively or passively, he would make an introduction: "My name is Mr. Blah. It is my honor to introduce my wife Mrs. Blah to you."
- Whenever he receives such an introduction from another gentleman, he passes the introduction to his wife.

Lady's Rule:

- Whenever she receives an introduction from her husband or another gentleman, if she does not know the introduced lady yet, she would connect with that lady and tell her who introduced her.
- Whenever she receives a connection request and was told who introduced her, she will accept the request only if the introduction was from her husband.

The above rules can make sure all the ladies and gentleman are properly connected. We refer to these rules as *Pair Connection Protocol*.

To implement the Pair Connection Protocol in Dewblock, one more type of message was added. This message type is ALTERNATE_ADDRESS. This type of message can travel in both inter-node channels and cloud-dew channels.

Whenever a dew client initiates a connection or receives a connection request, it sends out an ALTERNATE_ADDRESS message to the other end of the connection. Whenever a dew client receives such a message, it passes this message to its cloud server.

Whenever a cloud server or a full client receives an ALTERNATE_ADDRESS message, it first checks if such a connection already exists; if not, it initiates a connection with the address specified in the message and sends this message to the receiver. When a cloud server receives a connection request and an ALTERNATE_ADDRESS message, it verifies that the message was originated from this cloud server's dew server before it accepts the connection.

V. Collaboration Mechanisms

Let us continue to use the family analogy introduced in Section IV to describe Dewblock's collaboration mechanisms. These descriptions further reveal the features of Dewblock, and also demonstrate the important role of family analogy in inspiring and explaining these mechanisms.

A. Integrity Keeping

If a dew client can perform all cryptocurrency operations, why do we need a cloud server? In other words, why do we need to operate a full node? The following analogy provides an explanation.

All families in a community need to maintain the community's justice and well being. They need not only to work for their own families, but also to vote for the community for various reasons. Every family should have a voter. To satisfy this requirement, every family designates the lady of the family as the voter. Whenever a vote is called, the gentlemen would ignore the call, but the ladies would vote. In a community with strict rules, if a family does not participate voting for a while, this family shall be excluded from the community.

In Dewblock, we have a similar situation. Each node has its responsibility to keep the integrity of the blockchain network. Each node needs not only to operate the node's own functions, but also to keep the whole blockchain and provide the whole blockchain to other nodes when needed. The cloud server or the full client of each node are designated to fulfill this responsibility. The dew client of each node would ignore the request to provide the whole blockchain. Strictly speaking, If a node does not fulfill its responsibility in keeping the integrity of the blockchain network for a while, this node might be disconnected from the network. This exclusion rule has not been implemented in Dewblock code yet and is on our future agenda.

B. Mining in Cloud

Block mining is an important activity in blockchains. How is mining performed in Dewblock? Let us check the similar situation in the family analogy first.

All families in a community have meals together in a shared fashion. It is an honour for a family to cook for the community. Gentlemen can cook, but ladies cook better. Single ladies know when to cook, but wives only cook when their husbands ask them to do so.

Let us go back to Dewblock. Dew clients can perform block mining. Because mining takes huge amount of computing power, it may seriously influence the normal operation of a

personal computer or a mobile device where the dew client is running. A better arrangement would be to ask the cloud server to mine a new block on behalf of the dew client. A new kind of message, MINING_REQUEST, was introduced. This kind of message travels through cloud-dew channels. When such a message is received, the cloud server tries to mine a new block; if successful, the new block will be added to the Dewblock network.

VI. Dewblock Project

Dewblock is a blockchain cryptocurrency system that was developed as a proof-of-concept system for the principles described in this paper. It can be modified to accommodate transactions on records other than cryptocurrencies.

Dewblock was developed on top of an open source project Naivecoin [9][10]. Naivecoin is a blockchain cryptocurrency system. It tries to show that the basic principles in a cryptocurrency can be implemented in a concise way. Naivecoin uses the UTXO model.

Dewblock was developed through the following major changes to Naivecoin:

- Naivecoin's underlying state-keeping model has been changed from the UTXO model to the account model. Such modified Naivecoin was referred to as Account-Naivecoin in this paper.
- Cloud-dew architecture was introduced. Dewblock contains two packages: *Dewblock-cloud-server* and *Dewblock-dew-server*. These two packages are modified versions of Account-Naivecoin.
- Dew servers can operate in two different modes: dew mode and full mode. When a dew server is in dew mode, it operates without the blockchain.
- A new kind of message channel, cloud-dew channel, was added. Four new types of messages were added.
- Pair Connection Protocol was implemented.
- Collaboration mechanisms, such as integrity keeping and mining in cloud, were implemented.
- Web commands were updated; configuration files were added.

The source code of Dewblock is stored in Github [11] as open source software under Apache License. The implementation details and operation instructions can be found in http://www.dewblock.com.

The most significant feature of Dewblock is that it provides a dew client; the dew client does not keep blockchain data; the dew client is part of a blockchain full node. Such dew clients can be introduced in various blockchain systems and deployed to personal computers and mobile devices.

VII. Conclusions

At the heart of the blockchain technology, a distributed secure ledge, a blockchain, is stored in every node of the network and trust can be established among unknown parties. By definition, blockchain data has to exist in every node and the data amount increases with time. Thus, the problem we are trying to tackle, blockchain clients' data size is too big and always increasing, is inherent to this technology. In the past, some approaches have been proposed to reduce the data size of blockchain clients, but these clients do not have the status of full nodes. Although these approaches provide convenience to users, they cannot be used in the backbone of blockchain networks.

Dewblock brings in a new approach that the data size of a client is reduced and the features of a full node are still kept. The key point is that the two concepts, blockchain client and blockchain node, are not the same any more in Dewblock. While a client is light-weighted and is conveniently operated in a personal computer or a mobile device, the client works with a remote cloud server to act as a full node.

This approach was inspired by dew computing principles. The architecture of Dewblock is the cloud-dew architecture. A dew client operates independently to perform blockchain activities; it also collaborates with the cloud server to maintain the integrity of the whole blockchain network. Two major features of dew computing, independence and collaboration, are demonstrated clearly in this application.

With Dewblock, each blockchain user needs to deploy a cloud server to a cloud service. From technical and economical viewpoint, the widely use of cloud services by individual users, including blockchain users, is feasible and affordable. Apparently, Dewblock, as a dew computing application, promotes the usage of cloud computing. This fact demonstrates the relationship between dew computing and cloud computing: cloud computing enabled dew computing; dew computing further promotes cloud computing; dew computing is the complementary piece of cloud computing [12].

References

[1] Sthuthie Murthy. (2018, May) "Blockchain will do for transactions what the internet did for information" - says IBM CEO. [Online]. Available: https://ambcrypto.com/blockchain-for-transactions-internet-for-information-ibm-ceo/

[2] Yingwei Wang. (2018, Sept.) Dewblock. [Online]. Available: http://www.dewblock.com/

[3] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[4] Yingwei Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[5] Vitalik Buterin. (2013, Dec.) A next-generation smart contract and decentralized application platform. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum

[6] Satoshi Nakamoto. (2009, May) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[7] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2202–2303, 2016.

[8] Brian Curran. (2018, Jul.) Comparing bitcoin & ethereum: UTXO vs account based transaction models. [Online]. Available: https://blockonomi.com/utxo-vs-account-based-transaction-models/

[9] lhartikk. (2017, Dec.) Naivecoin: a tutorial for building a cryptocurrency. [Online]. Available: https://lhartikk.github.io/

[10] ——. (2017, Dec.) Naivecoin. [Online]. Available: https://github.com/lhartikk/naivecoin

[11] Yingwei Wang. (2018, Aug.) Dewblock. [Online]. Available: https://github.com/yingweiwang/dewblock

[12] A. Rindos and Y. Wang, "Dew computing: The complementary piece of cloud computing," in *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on*. IEEE, 2016, pp. 15–20.