Section-Blockchain: A Storage Reduced Blockchain Protocol, the Foundation of An Autotrophic Decentralized Storage Architecture

Yibin Xu
School of Computer Science and Informatics
Cardiff University
Cardiff, UK
work@xuyibin.top

***Abstract*—Bitcoin-derived blockchain has shown promise as infrastructure for many decentralized models. However, problems are withholding the realization of the potentials. The booming increase of storage demand has hindered devices with storage shortage to use blockchain powered distributed applications. In this paper, we present section-blockchain, a new blockchain protocol, which is designed to solve the oversize storage problem without compromise the security of blockchain. There are no full nodes or lightweight nodes, all nodes are equal and contributing to the Section-Blockchain network. Experiments demonstrate that Section-Blockchain is efficient, remarkably reduced the storage, and withstand sudden nodes losing of massive scale. Section-blockchain also extended the capability of blockchain to the foundation of an autotrophic, tamper-resist decentralized storage system, in which, data can be equidistributional distributed worldwide automatically without any centralized dispatcher to assign storage; nodes are motivated to change their local storage to receive more remuneration. The global storage distribution is continuously optimizing and fit into any adding/losing nodes.**

Keywords—Distributed Networks, Decentralization, Dynamic Storage Management, Blockchain, Distributed Ledger Technology

I. INTRODUCTION

In general, cryptocurrencies [2] (e.g. Bitcoin [1] and altercoins [4,8]) have earned much attention from fields like financing, technology and academic, achieved significant market penetration in underground economies [3], overall attracted near $12.4B in venture capital [4] and reached a market cap of $78.42 billion as the year of 2017 [5]. Its wide-deploy, has been resounding evidence that the underpinning technology –blockchain, as distributed ledger is practical and secure. Blockchain also promises to become the infrastructure for most decentralized models. It helps to reform the Internet interaction, improves the anonymous online payment systems, safeguards digital contracts and empowers anonymous parties to programmatically complied to complex agreements [7,8]. However, Blockchain is bulky, though it is only dozens of bytes per transaction, the overall storage demand for a full node to keep all the transactions has reached 165.50 GB as the year of 2017 [6] and is continuing to grow. In the foreseeable future, blockchain may face storage problem regarding home devices and mobile devices which have limited storage capacity. Though lightweight node is proposed [9], it compromised the anonymity because the full node can establish a link between wallet addresses and the IP addresses of whom sent transaction inquiries. Besides, the lightweight node is unable to determine if the full node has returned all related transactions. As the lightweight node is also unable to confirm blocks, the system is tending to be a centralized one once the number of full nodes dropped sharply. Though many implements of blockchain in IoT [12-15] achieves the scalability and storage reduced using the idea of lightweight nodes and authorized partly centralization, that just runs opposite the initial ideal of blockchain as anonymous, decentralized and highly secured. Thus, to solve the oversize program of blockchain is also to secure the blockchain system.

In section 3, we present Section-Blockchain, a storage reduced blockchain protocol. Section-Blockchain runs on an efficient communication protocol [11] which we will briefly discuss in section 2.2. This protocol helps the nodes to optimize their positions in the network as to achieve a formatted network layout and a much faster data broadcasting speed. We redesigned block storage mechanism base on original Nakamoto blockchain, allowing nodes only to keep (1) the block headers in the mainchain; (2) a certain number of blockchain fragments; (3) a certain number of Database Snapshot (DS); (4) a Fragment and Database snapshot Grabbing Routing Table (FDGRT). A fragment is a set of blocks in the mainchain; DS contains the status of records (e.g., the account balance at a particular moment and the global settings). Section-Blockchain is using the same Proof of work (PoW) as Nakamoto blockchain to select the winner block. However, block publishers need to additionally include the evidence of storing a sufficient number of fragment and DS into blocks. There is a compulsory bottom line of the number of fragments and DSs that any block publisher must save, which is autonomously adjusted based on the number and the download speed of copies of all fragments and DSs in the network. Nodes can join in mining when the number of fragments and DSs they stored is above the bottom line. Generally, the more fragments and DSs a miner saved, the more compensation the miner will gain; the rarer a fragment and paired DS is being stored globally, the more compensation this fragment and DS worth. Owing to the remuneration rule of Section-Blockchain (will be discussed in section 3.1), a node is tending to grab fragments and DSs that are rare in its network neighbourhood when it considers adding/change fragment and DS to store. This rule helped the data to be stored distributed and dynamic in the system. Experiments in later sections showed the efficiency, remarkable storage reducing and the robust of section-blockchain in a heterogeneous and unstable network situation.

The way, fragment and DS are preserved, provides a foundation for a large fault-tolerance and auto-trophic decentralized storage system. Section-blockchain can be an alternative for many existing distributed systems nowadays, e.g., Content Delivery Network, which requires the centralized dispatcher to assign storage globally. The existing models may be slow when the data is extensive and is frequently updated globally. And they are also vulnerable because one only needs to break down the dispatcher to break down the system. This article will discuss Section-

978-1-5386-9341-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICECCS2018.2018.00020

Blockchain protocol in detail but only provide an idea for the decentralized file storage system build above it.

II. PRELIMINARIES

It is assumed familiarity with Bitcoin, the procedure as well as the structure of the Nakamoto blockchain [1]. We will briefly describe the basic concepts of lightweight blockchain node and the new communication protocol first proposed in [11] in this section.

2.1. Lightweight Node

Lightweight nodes connect to Blockchain network but do not store transactions or validate blocks. We refer nodes which store the full copy of blocks as the full nodes, and lightweight are clients to the full nodes. Lightweight nodes only stone the complete copy of the block headers in the mainchain. Lightweight nodes linearly scaled with the height of the blockchain at only 80 bytes per block header, up to 4.2MB per year, regardless of the total size of blockchain [9]. Lightweight nodes can verify if transactions in question are embedded in certain blocks by the Merkle root in the block header along with a Merkle branch received from a full node.

2.2. Communication Protocol

It is predefined in our communication protocol that two nodes can become peers when they don't have a mutual peer, thus nodes must carefully calculate the tradeoff between the benefit received for connecting to a specific node and the connection restrictions after peering to this node. Data are sent in parts if it is larger than 1Kbytes, when all a node heard all the parts of the data, it marks the finish of a data propagation. The same as the original communication protocol, nodes will acquire several peer information from a DNS-like server at the beginning. Instead of connecting to them directly, our communication protocol uses several indicators to select some suitable nodes to connect.

When a node broadcast Data (transaction, block etc.) to the network, if the data is larger than 2 Kbytes, it is divided into parts, a part has a fixed size of 512 bytes. Before a data propagate begins, a data header of a tiny size is sent to the network, which indicates which type of data will be propagated (block header/block/ transaction) and the Merkle Root of that data. Then, the node sends the parts to its peers, each part with a Merkle branch that can derive the Merkle Root of this data. Different parts are being sent to different peers at the same moment, when all the parts are being sent out, this marks the end of a moment, the next moment will start over until every peer has heard the entire data. An example of this procedure is showed in Figure.1, where the data sender has four peers A, B, C and D.

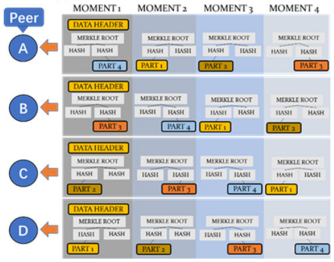

Fig. 1. Procedure of Publishing a Data to The Network

In the blockchain network, every node can send transactions/blocks to the network. The philosophy of this communication protocol is that for every node, generally, it can hear some pieces of data faster than some other nodes then retransmit the data earlier than some others because every node gets a chance to be the peer of the data publisher or closer to the data publisher than others. If the layout of the network is well organized, for every node, the number of data pieces that the node first heard from a peer in a fixed time window should be within a range which differentiated by a combined impact of several indexes. The protocol quantified the network structure and the performance of every node through the following indexes (supported the node we are operating is called node A):

(1) **NID (Node identity).** NID is a 32 bytes Ed25519 [20] public key. The owner of the private key paired to this public key is the owner of this identity. Unlike wallet address, this identity attached blank currency value, and is used only as the identity of a node.

(2) **IPC (Index of Peer Coincidence).** $IPC_{A,B}$ is the IPC between node A and node B from node A's perspective; $IPC_{A,B} \coloneqq \frac{\mathrm{Card}(\mathrm{SubPL}_B \backslash \mathrm{PL}_A)}{\mathrm{PN}_A}$,where SubPL_B is the set of all the peers of all the peers of node B, PL_A is the set of all the peers of node A, PN_A is the number of peers of node A. Figure.2 shows an example of the PL and SubPL of node A and node B. In this scenario, $IPC_{A,B} = \frac{3}{4}$.

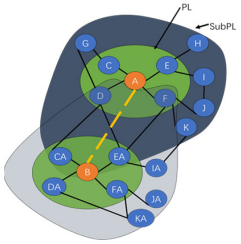

Fig. 2. An example of PL, SubPL and IPC

(3) **ND (Network Distance).** $ND_{A,B}$ is the ND between node A and node B, which is defined as the time in MS consumed for A to retrieve a data that is sized 1 Mbytes from B.

(4) **SP (Structure Proportion).** The structure proportion of node B from node A's perspective is $SP_{A,B} \coloneqq \mathrm{PN}_B \times IPC_{A,B} \times (1 + \frac{ND_{A,B}}{1000})$.

(5) **SI (Structure Index).** $SI_A \coloneqq \frac{\sum_{i=1}^{i=\mathrm{PN}_A} SP_{A,i}}{\mathrm{PN}_A}$.

(6) **NFHDP (Number of First Heard Data Part in one data Propagation).** $NFHDP_B$is the number of parts of a data that is first heard from peer B in a data propagation.

(7) **RT (Real Time).** The time in second since the system was started.

(8) **ExpNFHDP.** When node A finished hearing data from one data propagation, it creates a set of arrays ODP, $\mathrm{ODP}_{-\infty\ldots+\infty,-\infty\ldots+\infty} \coloneqq \emptyset$ is the initial value, then $\mathrm{ODP}_{\mathrm{SP_{A,i}},\mathrm{NFHDP_i}} \coloneqq \mathrm{NFHDP_i}$, $\mathrm{i} \in [1, \mathrm{PN_A}]$ .The ExpNFHDP for its peers are: $\mathrm{Exp}NFHDP_i \coloneqq Average\left(\mathrm{Grubbs\ criterion}\left(\mathrm{EDP}_{SP_{A,i},NFHDP_i}\right)\right), i \in [1, \mathrm{PN}_A]$, where $\mathrm{EDP}_{SP_{A,i},NFHDP_i} = \{\mathrm{ODP}_{k,j}\}$, $0 \le j \le$

$Max(NFHDP_{1..PN_A}), SP_{A,i} - 0.5 \le k \le SP_{A,i} + 0.5$, Grubbs criterion is a function that returns a set of data which has eliminated abnormal data from the original set of data. Specially, if the data propagated was not split into parts, Exp$NFHDP_i := 1$.

Figure.3 shows an example of this procedure, where the Exp$NFHDP$ for the red dot is the result of Grubbs criterion of the dots in yellow rectangle.

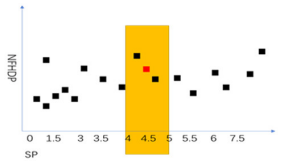

Fig. 3. An example of **Exp$NFHDP$**

(9) FR (Fulfil Rate): $FR_{A,B,E} := \sin\left(\min\left(\frac{3}{2}\pi, \mathrm{P} \times \frac{\pi}{2}\right)\right), P := \frac{\sum_{j=0}^{j=PN_A} \sum_{i=1}^{i=N_{B,E}} \frac{NFHDP_j^i + 1}{\mathrm{Exp}NFHDP_j^i + 1}}{PN_A}$, where $N_{B,E}$ is the number of data propagations during RT time B to E, Exp$NFHDP_j$ is the expected number of $NFHDP_j$; $NFHDP_j^i$ and Exp$NFHDP_j^i$ is $NFHDP_j$ and Exp$NFHDP_j$ at i data propagation respectively.

(10) AB (Average Bandwidth): $AB_{B,E} := \frac{size(data_{B,E})}{E-B}$, where $\boldsymbol{data_{B,E}}$ is the set of data received between RT time B to E; size(T) is the size of T in Kbytes.

When two nodes become peers, they will co-sign a contract stating a minimum $SP_{A,B}$ and a minimum $SP_{B,A}$ that each side of the connection must maintain, and the time of this contract. The signed contract will be sent to their existing peers so that relevant indexes from some other peers' perspective can be recalculated. After a data propagation, a node must send a signed work confirm to peers which sent pieces of data to it in this data propagation. The contract can be terminated prior to its expiry time; if both sides of the contract agree to terminate, they will exchange a signed terminate notice, and then send the termination notice signed by the other side of the connection to their other peers. However, if only one node (a node A) agrees to terminate (with a node B), node A will stop retransmit data to node B and stop sending work confirms. Then, after three data propagations, node A informs its other peers about this termination with node B. Others will check on node B to see if it can provide a work confirm signed by node A that is among the three most recent data propagation. Then this termination is invalid; Otherwise, they will deem this connection as terminated and recalculate the indexes. When a node not receiving two continuous work confirms which it supposes to receive, it will start the connection termination process introduced above.

A reinforcement learning model is used for every node A to automatically explore a maximum benefit from adjusting its peer structure as well as supervise each other's behaviour. We say this model is tuple:

- State $:= (\mathrm{FR}_{X,c-W,c}, Average(SI_X^{(c-W)\ldots c}))$
- Action $:= \{\mathrm{ADD}, \mathrm{DELETE}, \mathrm{STAY}\}$,
- Reward $:= \mathrm{AB}_{c-W,c} - \mathrm{AB}_{c-2\times W,c-W}$,
- Policy

where SI_A^i is the SI_A at RT time i , W is a fixed intended time window for this algorithm to function, c is the current RT time, $c \bmod W = 0$; Action ADD is a function that selects and builds contract connection with a node from a peer candidate pool by calculating a dynamic programming formula using the current PL_A as initiative state (see [11] for details). Action DELETE is a function that terminates the contract with the least valuable peer. Action STAY is a function which do nothing. The policy is calculated automatically by the reinforcement learning algorithm.

This communication protocol ensures that the network will not be a strongly connected network or have many redundancy connections. Thus, it provides the effectiveness of jump distance (will be defined in section 3) to evaluate the distribution of storage.

III. SECTION-BLOCKCHAIN

We outline Section-blockchain through section 3.1; discuss the constrains and motivations for individual participants that functions the section-blockchain in section 3.2; unfold some details of section 3.1 in the remaining parts of section 3. Figure.9, a picture illustration of working procedure of an individual node is given at the end of section 3.1.

3.1 Definitions

(1) Fragment and DS (Database Snapshot). A fragment contains a fixed number of blocks, and DS is the states of blockchain in a particular moment. Every time a fragment is formed, the transactions inside this fragment is executed in sequence, the results are written into a new DS which updated from the preceding DS. A fragment and the preceding DS are paired, and they are always stored together. Throughout description of Fragment and DS is given in section 3.2. Figure.4. shows the relationship among blocks in mainchain, fragment and database snapshot. In this scenario, a fragment stores 4 blocks.

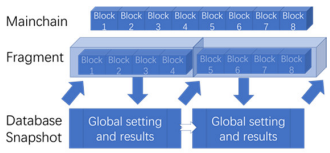

Fig. 4. Mainchain, fragment, DS relationship

(2) JD (Jump distance) from any node A to any node B is defined as the number of nodes passed in the ND path from A to B

(3) ND (Network distance) is the shortest time required for A to retrieve a fragment and paired DS from B.

(4) T/G/∂ Section-Blockchain is a section-blockchain system that fulfilled the following conditions:

1) Every fragment has T continuous blocks inside;

2) For every fragment and the preceding DS, every node records G number of paths to different sources of this fragment and the paired DS. Which sources are, to the node's knowledge, of the shortest NDs with this node;

3) Every node should be able to retrieve every fragment and paired DS with an average JD (Jump Distance) of G paths that is near ∂.

(5) Longest average jump distance β is a value which is written in the block header of every block. Supported node i published a block, the β in the block header is:

$\beta \coloneqq MAX\left(\frac{\mathrm{SUM}(\mathrm{JD}_i^k Fragment\ j)}{G} \middle| k \in [1, G], \mathrm{F}: \cup_j Fragment\ j\right)$,where F is the set of all fragment,$\mathrm{JD}_i^k Fragment\ j$ is the number k shortest jump distance for node i to retrieve $Fragment\ j$ as well as the paired database snapshot. For example, in Figure.5, let's say node i is the red cylindricity, other cylindricities are other nodes in the network. If there is a blue edge between two cylindricities, these two nodes are connected. Supported $G = 4$ and there are two fragments and paired DSs ever existed in the network which represented by the blue stars and smelling faces respectively. β is 7/4 (The sum of JD for the blue star is 7 while the sum of JD for the smelling face is 5).

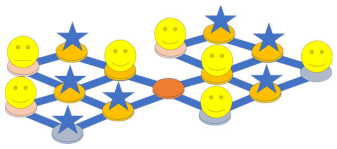

Fig. 5. An example of β

(6) The Bottom line of storing. For every fragment, θ is the median value of all β indicated in blocks in that fragment. θ suggests the median largest grabbing distance for the whole network at the time when the blocks in this fragment are generated. We say θ_g is the θ at g fragment, B_g is the bottom line for storing fragments in the period of g fragment generation phase, $B_g \coloneqq CEIL(AVERAGE(B_{g-1} \times \frac{\partial}{\theta_{g-1}}, B_{g-1})), B_0 = 1, B_1 = 1, \theta_0 = \partial, \theta_1 = \partial$. ($\partial$ is defined in (4)).

(7) ℵ. The block publisher indicates the evidence of storing specific fragment and paired DS into the block. We will discuss the evidence in section 3.4. When a block contains the evidence of storing a fragment and the paired DS, this fragment and DS is being claimed. $\aleph(j)$ is the internal time for fragment j and the paired DS to be claimed: supported current block height is BH, the block height of last time fragment j was claimed is BH_{last}, $\aleph(j) \coloneqq BH - BH_{last}$. $MAX\aleph \coloneqq MAX(\aleph(j)), j \in F$ (F is defined in (5)). Figure.6. shows an example where there are nine fragment and DS in the system. Block X carried the evidence of storing fragment & DS 2 and fragment & DS 5. When Block X is accepted into the blockchain, the value of these fragment and DS from ℵ list returned zero while the value of other fragment and DS added one.

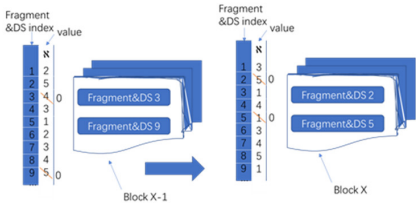

Fig. 6. An example of ℵ

(8) Compulsory storage. The Block publisher in the period of g fragment generation needs to store at least B_g numbers of fragments and the paired DSs; for every $C, C \in [1, B_g]$, the miner should store at least one fragment j which $\aleph(j) \in \left(\frac{MAX\aleph}{B_g} \times (C-1), \frac{MAX\aleph}{B_g} \times C\right]$ .Figure.7. shows an example, where $B_g = 2$, $MAX\aleph = 5$. There are nine fragments and DSs in the network, the ℵ list is rearranged by the value of ℵ, and the miners are required to store one from each window divided by C ($\left(\frac{5}{2} \times (0), \frac{5}{2} \times 1\right]$ and $\left(\frac{5}{2} \times (1), \frac{5}{2} \times 2\right]$).

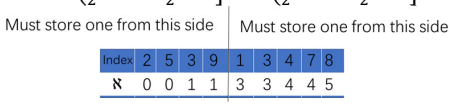

Fig. 7. An example of compulsory storage

(9) Reward. The reward for keeping a fragment j is $Reward(j) \coloneqq C \times k$, where $\aleph(j) \in \left(\frac{MAX\aleph}{B_g} \times (C-1), \frac{MAX\aleph}{B_g} \times C\right]$ and $k = \frac{0.5\varepsilon}{(B_g\sim)}$,ε is the maximum reward for storing at current block height. For example, if $B_g = 2, \varepsilon = 9, k = \frac{4.5}{2+1+0} = 1.5$. Figure.8. shows the reward R for store a fragment and paired DS using the data in Figure.7.

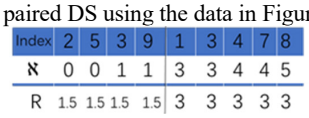

Fig. 8. An example of R

K ensures that when the block publisher kept the bottom line of fragments and DSs, the publisher will receive 50% of the maximum reward at that block height.

(10) Mining Reward. The reward for a block publisher who stored i number of fragments is $\min(\sum Reward(j), \varepsilon), j \in \omega$, where ω is the set of fragments it stored, ε is the maximum reward for storing at the current block height. The storage strategy will be discussed in section 3.6.

(11) Guarantee nodes. There are G (the same G as indicated in (4)) numbers of nodes selected as guarantee nodes per fragment and paired DS. Selected nodes should continuously keep this fragment and paired DS until a fixed number of continuous blocks since the selection has been finally accepted by the network (in this paper we say this fixed number is 5000). Penalty will be imposed if nodes violate this rule. After duty time, the fulfilled nodes will receive ε amount of remuneration. The details of the guarantee node are presented in section 3.7.

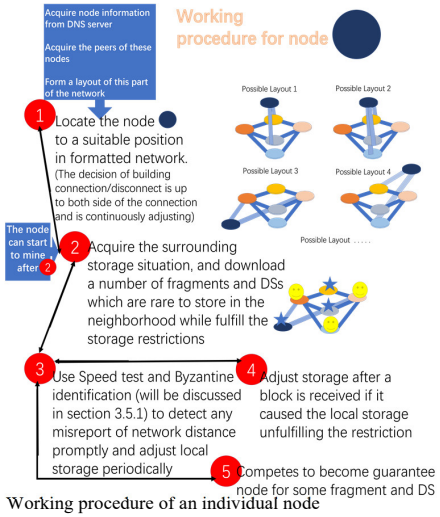

Fig. 9. Working procedure of an individual node

3.2 Constraint and Motivation

a) The constraint of peering, a motivation of network adjustment

Our communication protocol has prevented nodes from connecting too many nodes; nodes are motivated to adjust their peers as to seek a better network position and then have an advantage in hearing data. Since two nodes can become peer only when they don't have a mutual peer, nodes are motivated to compare the benefit and drawback from connecting to/ disconnect a node. Nodes must monitor the preference of others as to prevent themselves from being disconnected by others. Thus, the constraint of peering motivated the global network structure adjustment.

b) The constraint of storage choosing, a motivation of global autonomous distribution of storage

The requirement of storing the bottom line of fragment and DS as well as the restriction of what fragment and DS a node must store have compelled the nodes to adjust their local storage frequently for creating valid blocks. It is natural for nodes to store data which are less affluent in number in its neighborhood to achieve a less adjustment expectation. When nodes globally followed this rule, the data is autonomously distributed equally without any central controls.

c) Local optimize for global optimize

It is natural from the financial perspective for nodes to download high price data after fulfilled the storage restrictions. The pricing for fragment and DS gives nodes a global view of the affluence of specific fragments and DSs. If the price is high, it means this fragment and DS hasn't been claimed for a long time, which implies there are few nodes stored this fragment recently. Thus, additionally keeping high price fragments and DSs, from the local point, the financial reward is maximized, while from the global point, it strengthened the equal division of storage.

d) Limit reward for less data losing expectation

When a block publisher stored the bottom line of data, it is guaranteed a 50% of the current highest reward. When the value of the fragment and data added up exceeded the maximum reward, the block publisher will not receive additional rewards. This restriction incents nodes to adjust their storage frequently instead of store massive data. And this also safeguard the system from losing some data when a node with massive data go offline unexpectedly.

e) Guarantee node as anchor and diffusion central

Since Guarantee nodes must store specific fragment and DS for a fixed continuous block interval, it prevented the accidental data loss in unordered autonomous storage adjust (e.g., all sources of a data drop this data simultaneously, though the chance of this is scarce), thus guarantee nodes are the security anchors. Besides, guarantee nodes are also diffusion centers. While it keeps this particular fragment and DS, other nodes may drop this fragment and DS for financial purpose or due to the affluence of this fragment and DS in the neighborhood, the storage after autonomous adjustment of sometimes is distributed ripples around the guarantee nodes. Thus, guarantee nodes latently guided the distribution of the storage of specific fragment and DS. Figure.10. shows the example of the distribution, where red cylinders are guarantee nodes for the blue star, other sources of the blue star are in the ripples of the guarantee nodes.

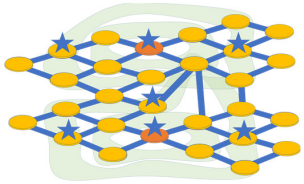

Fig. 10. An example of data distribution

3.3 Fragments, Database Snapshot

When a node has a fragment and the paired DS, it is a source of this fragment and DS. Every fragment consists of a chain of blocks, fragments are also chained. A fragment is finally accepted when the last block in it got α confirms from the network where α is the globally recognized confirm-time for finally accept a block. If placing all the fragments in sequence, the chain of blocks in them is identical to the finally accepted blocks in the mainchain of blockchain in both content and sequence. Fragment is proposed for two proposes: (1) globally pre-standardize the window for storing; (2) derivate DS with less forking. A block can only belong to one final accepted fragment. Every time there are T number of contiguous blocks in mainchain that has not been included in a final accepted fragment, a new fragment will be created with these T continuous blocks in it. The transactions inside this fragment are executed in sequence, the results are written into a new DS based on the preceding DS. After deriving a new final accept fragment and the new DS, if nodes are not planning to hold this final accepted fragment and the paired (preceding) DS, they will delete all the blocks in this fragment as well as the paired DS but remain the new DS and the block header of those blocks. Figure.11. shows the relationship of fragment and DS in a Section-Blockchain system in which $T = 4, \alpha = 3$. Figure.12. shows the working procedure of fragment generation and DS update in this system.

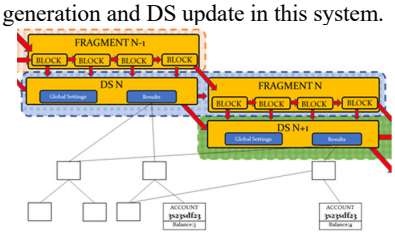

Fig. 11. The relationship of fragment and DS

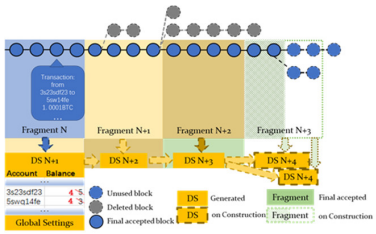

Fig. 12. Procedure of Fragment and DS

If a final accepted fragment of a forked chain and the DS it derived emerged in the network (the probability of this is tiny), it will soon be detected through the difference of the Merkle Root of fragment and DS indicated in the block header in the coming blocks, the procedure to deal with this situation is:
(1) Retrieve all fragments in that unknown chain since forking as well as the last DS before the forking from the network

(The Merkle Root of fragment and DS retrieved must be identical to the ones indicated in block headers of that period);
(2) Verify and execute transactions in fragments retrieved base on the DS retrieved until the newest DS comes out. The Merkle Root of this fragment and DS should be identical to the one indicated in the block in question;
(3) Compare the amount of Proof of Work (POW) and shift to the chain which required more work.
DS of any period can be regenerated at any time by retrieving relevant fragments and DSs from the network. Thus, referring DS and its following blocks to verify coming transactions is as secure as using the original procedure which looking up relevant transactions in previous blocks of the blockchain.

3.4 Block, Proof of Storage and Supporting material

We use segregate witness [10] to raise the transaction capacity per block. Block in section-blockchain is composed of a block header, transactions (together sized maximum 1MB per block) and a Huffman Code compressed supporting material (will be introduced in section 3.4.2). Besides standard information inherited from Nakamoto blockchain, the block header in Section-Blockchain has additional details of Merkle root of preceding fragment and DS, the number of items in preceding DS, Merkle root of supporting material, number of Merkle leaves of supporting material and β (longest average jump distance). Figure.13. shows the structure of block in Section-Blockchain.

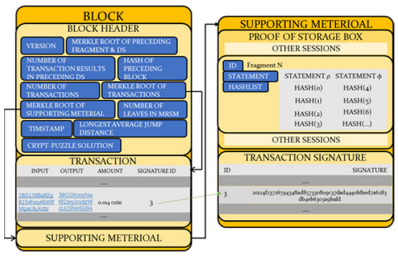

Fig. 13. Block Structure

3.4.1 Proof of Storage

For every fragment N and its preceding DS, Fs is the set of information of every block (including supporting material) inside this fragment and every piece of information in its preceding DS. Place the information and supporting material block by block and information in the preceding DS linearly, a unified sequence number starts from 1 can be given. In this section, we refer content ρ as the sequence number ρ of the content in this fragment. Fs: $\bigcup_\rho contentp$. In Section-Blockchain, one is considered truly owns a fragment and the preceding DS, when and only when it can provide the exact $contentφ$ and $contentp$; φ and ρ are any sequence numbers which range from 1 to the total number of contents in Fs, $\varphi \leq \rho$. φ and ρ is given by the inquirer. PoS(N, φ, ρ) represents a request for proving storing fragment N and preceding DS, using $contentφ$ and $contentp$. The interrogee should return exact $contentφ$ and $contentp$ in compressed form and a list of hash named $HASHLIST$. The hash of decompressed $contentφ$, as well as the hash of decompressed $contentp$ and the hashes in $HASHLIST$ together, should be able to form the Merkle Root of block or DS which the contents belong to. To prove validation of $contentp$ and $contentφ$, the following is the proving procedure:

a) transform ρ and φ (the index number in fragment) to ρ' and φ' (the index number in blocks or DS which $contentp$ and $contentφ$ were written in) (the transaction numbers, the number of Merkle leaves of supporting material, and the number of information in DS are written in every block header, which helps this transform). we say the block(s) or DS which has $contentp$ and/or $contentφ$ inside are i and j respectively, i and j may be equal).
b) transform ρ' and φ' to Merkle tree index number $\rho''$and φ'' (the Merkle tree index of the Merkle Root of i and j are set to be 1; provided the index of a node in Merkle tree is A, for every A, its children 's index numbers are $2A$ and $2A+1$).
c) $result_\varphi = \text{decompress}(contentφ), result_\rho = \text{decompress}(contentp)$
d) if ρ equals to φ then $\rho'' = -1$
e) if $\rho'' > 1$ then goto(f) otherwise goto(h)
f) if ρ'' is an odd number and $\varphi'' = \rho'' - 1$ then $result_\rho := HASH(result_\varphi + result_\rho)$, $\rho'' := -1$, goto (j); otherwise goto (g)
g) if ρ'' is an odd number then $result_\rho := HASH(HASHLIST.pop(\ ) + result_\rho)$ otherwise $result_\rho := HASH\left(result_\rho + HASHLIST.pop(\ )\right)$.
h) if $\varphi'' > 1$ then goto(i) otherwise goto(j)
i) if φ'' is an odd number then $result_\varphi := HASH(HASHLIST.pop(\ ) + result_\varphi)$ otherwise $result_\varphi := HASH\left(result_\varphi + HASHLIST.pop(\ )\right)$.
j) if $\rho'' \leq 1$ and $\varphi'' \leq 1$ then goto(k) otherwise $\rho'' := int(\rho'' \div 2)$, $\varphi'' := int(\varphi' \div 2)$, goto (e).
k) if $\rho'' > 0$ goto(l) otherwise if $result_\varphi = M(k) = M(j)$, the result of PoS is valid.
l) if $result_\varphi = M(k)$ and $result_\rho = M(j)$, the result of PoS is valid.

We refer the Merkle Root of i as $M(i)$, j as $M(j)$, $HASHLIST.pop(\ )$ is a function that returns the content in the lowest index of $HASHLIST$. After a $HASHLIST.pop(\ )$, the content returned will be deleted in $HASHLIST$. Figure.14. is a visualized of proof of storage; supported $contentp$ is a transaction in Block i and $contentφ$ is a transaction signature in Block j, the gray rectangles are the information in $HASHLIST$. The orange rectangles are information in the path that is extracted from its child nodes. Green rectangles are information which every node knows. T(i) is the Merkle Root of transaction Merkle tree of block i. P(i) is the Merkle Root of Proving material of block i. S(i) is the Merkle Root of Signature Merkle Root of block i; PSB(i) is the Proof of Storage Box of block i (PSB will be discussed in section 3.4.2).

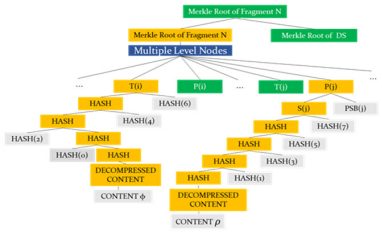

Fig. 14. Procedure of Proof of Storage

3.4.2 Supporting material

Supporting material consists of transaction signatures and Prove of Storage Box (PSB). PSB is a container, a section of PSB contains a fragment ID and the answer to PoS(N, φ, ρ) where

$$\varphi = MIN\begin{pmatrix}\text{BLOCKHASH}(i-1)\ xor\ \text{MT}(i-1)\ mod\ CARD(Fs_{Fragment\ N}),\\ \text{BLOCKHASH}(i-1)\ and\ \text{MT}(i-1)\ mod\ CARD(Fs_{Fragment\ N})\end{pmatrix}$$

$$\rho = MAX\begin{pmatrix} BLOCKHASH(i-1)\ xor\ MT(i-1)\ mod\ CARD(Fs_{Fragment\ N}), \\ BLOCKHASH(i-1)\ and\ MT(i-1)\ mod\ CARD(Fs_{Fragment\ N}) \end{pmatrix},$$

provided N is fragment ID, the block which this PSB belongs to is i, the proceeding block is $i-1$; BLOCKHASH(i) is the hash of block i that are transformed into decimal; MT(i) is the Merkle Root of transaction of block i that are transformed into decimal; *and* and *xor* are bitwise operations; Fs and PoS is defined in section 3.4.1. For blocks to be valid, blocks should indicate enough number of the valid evidence in PSB. Blocks should possess the Merkle Root of newest generated fragment and DS. We say the block(s) and (or) DS where content φ and content ρ are located is appointed by the network, which cannot be involved in Speed test and Byzantine Path identification of this period (will be discussed in 3.5.1).

3.5 Fragment and Database Snapshot Grabbing Routing Table

If a node A wants to obtain a fragment and the paired DS or sent a PoS enquiry, it will ask the peer indicated in FDGRT which is in the shortest network distance path to this fragment and the paired DS to grab them for it; the peer will continue to ask the peer's peer until reached the source. After receiving the fragment and DS, node A will check the Merkle Root of this fragment and DS received with the one indicated in the relevant block headers. If node A intended to store this fragment and DS, node A would inform its peers about the update. Nodes will change the record in their FDGRT, when a shorter path appeared, or a path is broken. The structure of FDGRT is simple, for every fragment and paired DS, it contains:

a) the ID of this fragment and the paired DS;
b) G numbers of shortest network distances to different sources of this fragment and the paired DS;
c) the node identities of these G sources;
d) the jump distances of these G paths;
e) the first peers passed in these G paths.
f) The node identities of the G numbers of guarantee nodes
g) The first peers passed in the paths to the guarantee nodes

Figure.15. shows a section-blockchain network of 4 nodes and a node A 's FDGRT. In this network G=2, and every node is the guarantee node for the fragment and DS it stored; the fragment and DS a node stored are shown alongside the node's name in the figure, the number on the edge is the network distance between the two sides of the edge.

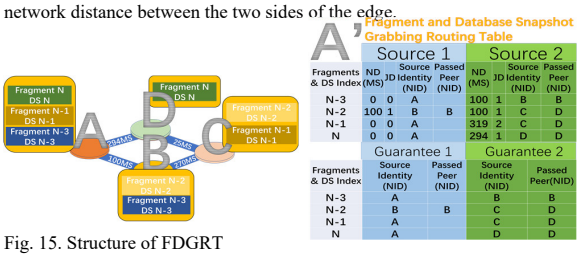

Fig. 15. Structure of FDGRT

3.5.1 Speed test and Byzantine identification

The protocol designs a Speed test and Byzantine identification agreement to question the genuineness of a path as well as the ND of specific fragment and DS provided by peers. It works as follows:

(1) When node A join in a path, node A should send PoS inspection to the source of the path through this path and calculate the time of response.
(2) Only when the return from the source of a path fulfill the proving criterion mentioned in section 3.4.1 and the time is match to the ND in FDGRT (the time is 120% above of ND in FDGRT or is below 80% of ND in FDGRT), this path is considered reliable otherwise the path should be deleted.
(3) Every node should conduct a random inspection to at least a path per peer at a time within every fragment forming period (including guarantee sources). A path can be any path that is passing through this peer.
(4) If the source of the path returned a wrong answer of PoS or not returning an answer, this path is invalid. All nodes in this path will seek to replace this path, and question other paths involved this source node.
(5) If node A is not receiving any PoS request from a peer (if there is one or more path involved node A and that peer) at a fragment formation period, it suggests this peer is not carrying the duty of inspection, this peer should be disconnected.

In Speed test and Byzantine identification, PoS request is PoS(N, φ, φ), where N is the ID of the fragment, φ is a random index number that complies with $\varphi \equiv MT(i)(mod\ (BLOCKHASH(i)\ mod\ Fs))$, i is the newest block in mainchain, Fs is the number of elements in fragment N and paired DS. Specially, $content\varphi$ cannot belong to the block or DS which is appointed by the network. Because every node randomly picks paths to inspect, every inspection is using random φ, the time for the inspection is also varied (anytime within a fragment period), a fraud path will not affect the network for a long time due to lacking inspection answers, or the speed is too slow if the Byzantine node asks others for the answer.

3.6 Grabbing Strategy

From every miner's perspective, to maximize the profit, the most street forward way is to greedily maximum every $\aleph(j), j \in \omega$, while maintaining the restriction of fragment and DS storing which described in section 3.1. Since $\aleph$ can be changed dramatically when a block emerged, and miners might need to download other fragments and DSs to fulfill the criteria of publishing blocks before validly join in the next mining, to make this happened less frequently, we want the storage map in the neighborhood to be unique. It is easy to prove that the more unique fragments and DSs nodes in the neighborhood kept, the less reward damage expectation and change-storage expectation. This method also helps fragments and DSs to be stored distributed and dynamically in the network, which brings the security when a sizeable geological zone of nodes lost connection in a second. The default fragment grabbing strategy is:

(1) 50%-50% chance for keeping a new generated final accepted fragment and its preceding DS.
(2) Hold up to twice the number of the bottom line of fragments and their preceding DS.
(3) When the storage doesn't meet the storage restriction, sync the closest fragment and paired DS in network distance which will make the storage meet the restriction.
(4) Set a random time window and conduct the following procedures every time of this time window: calculate the median of the jump distance of all the fragments and paired DSs. Replace the fragments and DSs currently hold that is below average.
(5) Set a different random time window and conduct the following procedures every time of this time window: calculate the median of the network distance of all the fragments and paired DSs. Replace the fragments and DSs currently hold that is below average.

(6) If the number of the data stored exceeded the upper limit the user set (since the maximum remuneration is ε, there is of no meaning for nodes to store a lot more if the price of all the fragment and DS it stored together has far exceeded ε), drop the data with the lowest median network distances while meeting the storage restriction. If there are several dropping options, choose the ones the node kept for the longest times.

3.7 Guarantee node

If a node wants to become a guarantee node, this node needs to send a frozen transaction (guarantee deposit) which contains the INPUT transaction, the hash of the latest block and the node identity (NID). When it is at the last block before a guarantee expired, miners will select the one who sent the highest amount of guarantee deposit as the replacement node for the expiring guarantee node and writes this guarantee deposit into the block. The guarantee deposit must be sent between the block height interval of the latest block and the preceding block of the latest block when the block is creating by its publisher. The selected frozen transaction and the fragment and DS index will be written into the block as to confirm the replacement of guarantee node. If X number of guarantees expired at the same block height, miners would select X number of highest guarantee deposits, and randomly pair them to the fragment and DSs which needs new guarantee nodes. If a node becomes a guarantee node for a fragment and DS, its frozen transaction cannot be used within its duty time. The remuneration will be given at the end of the duration.

For the new generated fragment and DS, its guarantee nodes are selected within the blocks of next continuous fragment; every block will recruit ceil($\frac{G}{T}$) numbers of guarantee nodes for this new generated fragment and DS until reaching G numbers of guarantee nodes. For example, if G=2, T=4, when a new fragment and DS is created, and the block height of the last block in this fragment is X, one of its guarantee nodes will be selected in block height X+1 while another one will be selected in block height X+2.

3.7.1 Lose and replace Guarantee node

Supported node A is doing its routine speed test and Byzantine identification procedure mentioned in section 3.5.1; she sent a PoS to a guarantee node(the red cylinder on the top in Figure.16), the enquiry travels followed the red arrow in Figure.16; However, the last jump before the guarantee node cannot get a response from the guarantee node; then this last jump reply the path that the source cannot be reached. Then, all nodes in the path asks their other peers to find this guarantee node for the answer of PoS (the green arrow in Figure.16); if this specific guarantee node cannot be reached eventually, then this guarantee node is lost. All the nodes that is involved will mark this guarantee node as violated. And the violation information will be written into the coming block and the new guarantee node selected will be present at that block.

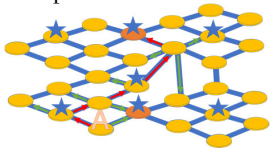

Fig. 16. An example of replacing guarantee node

IV. EXPERIMENTS

The purpose of the experiment is to show the storage required for individual nodes in section-blockchain and to test how robust section-blockchain is when facing nodes dropping of large scale within seconds. We do not make any comparison with other famous blockchain systems because they do not have protocols regarding saving storage for full nodes; their full nodes store all the data in the mainchain. The storage of these blockchains can be estimated without experiment since the size of the block is a fixed number, and a block interval is also a steady number. Also, there are no concerns about data loss in ordinary blockchains.

We use an emulated network with 2,000 Section-Blockchain nodes. 2,000 ubuntu systems are running as VPS (Virtual Private Server) on eight HP ProLiant SL230s Gen8 server; each VPS runs a node of Section-Blockchain. Hereinafter, we implement all the elements of Section-Blockchain introduced above to these emulated nodes. We give every VPS a randomly fixed upload bandwidth speed ranges from 400Kbit/s (50Kbyte/s) to 34,400Kbits/s (4.3Mbyte/s) and a fixed download bandwidth speed of 1600Kbit/s (200Kbyte/s) to 34,400Kbits(4.3Mbyte/s). The overall number of contract connections is 35742 among these 2000 nodes; The average number of contract connections a node has is 18; The least one peered with four neighbors while the most one peered with 99 nodes. All connections are given the network delay time range from 10ms to 600ms. Figure.17. shows the basic statistics of the simulated Section-Blockchain network.

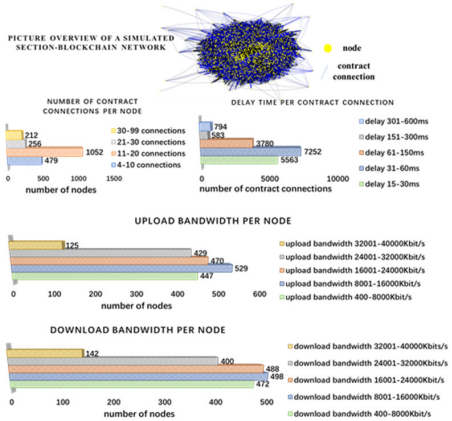

Fig. 17. Basic statistics of the simulated network

The reason for using these network setups is to simulate nodes in a various network situation. The communication protocol uses these network setups to form a well-arranged network layout, which layout is possible to exist in the real world. Since we do not test the time for data broadcast, the chosen of these numbers can be arbitrary as long as the numbers are varied.

4.1 Simulated Mining and The Bottom line of storing

The simulated mining experiment runs a 4/2/3 Section-Blockchain system with the block internal time of 1 minute; the experiment lasted two weeks. The purpose of this experiment is to show the reduction of storage and how the system is withstanding nodes dropping. We randomly assign memory and CPU frequency to each VPS as to simulate nodes

with different calculation ability. All nodes only store the bottom line number of fragments & DSs. For every two hours, there will be up to 20% of nodes suspended by the virtual switch as to simulate the unstable network. These nodes will connect to the network again within 6 hours. To make the experiment tougher, the suspending of nodes is sudden and simultaneous. For every node, it has 50% chance to use a new wallet address as Coinbase when publishing a block; within every five continue blocks generated, it has 50% chance to conduct a random transfer of a random amount of input that is within its balance to a random account indicated DS. When an account has zero balance, it will not be recorded in next DS; The transactions are of identical size. Figure.20. shows the fluctuation of online node number every minute (compared to one minute ago) during the experiment; Figure.18. shows the overall data generated during the experiment, this is also the scale of data that individual nodes in an ordinary blockchain would store since they don't distribute fragments. Figure.21. shows the average data a node of section-blockchain stored. Figure.19. shows the change of the bottom line of storing. Figure.22. shows the average percentage of data a node stored to negative one compared to the overall data generated. As the result in Figure.22 suggests, a node only needs to store 0.2% (0.2%-$1=\frac{1000}{2}=500$) of the overall storage in average with the fact that the system went through several times of 20% nodes lose in a sudden without losing any data. In a network system with thousands of nodes, it is almost impossible for it to lose 20% of its online users in a sudden unless a major network damage globally occurred. Thus, we believe offline 20% of nodes in a sudden is enough to show the robust of the system.

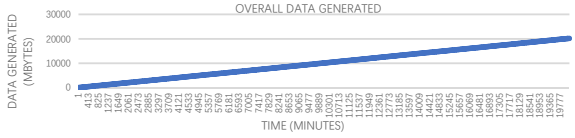

Fig. 18. The overall data generated

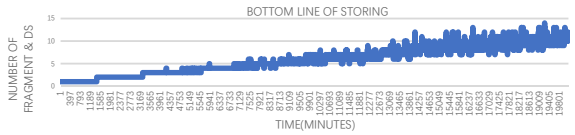

Fig. 19. The changes of the bottom line of storing

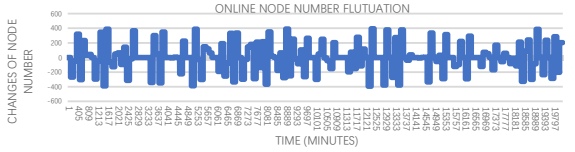

Fig. 20. The change of number of nodes online

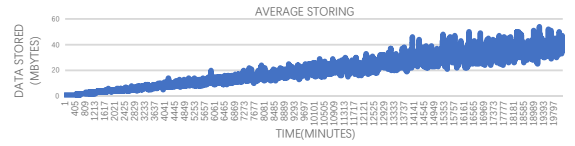

Fig. 21. Average size of data stored per node

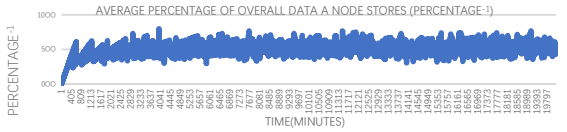

Fig. 22. Average percentage^{-1} of overall data a node stored

V. SECURITY ANALYSIS

5.1 *Possibility of Losing data*

Provided the overall number of different fragment and DS in system is F; there are N number of nodes which stored at less bottom line of data; if K, $K \leq N$ numbers of nodes suddenly went offline in this moment, the chance for a fragment and paired DS which has X number of sources in the network to lose all X numbers of sources: (1) When $X \geq G > K$, the opportunity is zero; (2) When $K \geq X \geq G$, the opportunity is $P := \frac{\binom{N-K}{K-X}}{\binom{N}{K}}$, when $K \geq X = G$, the chance reached the peak point. Provided there is F number of the different fragment and DS in the network, XA is the average X in the system, the average overall data losing expectation is $P := \frac{\binom{N-KA}{K-XA}}{\binom{N}{K}} \times F$.

Figure.23. shows the average overall data losing expectations using experiment data in section 4, when the system is experiencing an unstop, up to 20% node losing and adding. During the experiment, the highest expectation for losing data is near 4/1000, while mostly the expectations are under 1/1000. Though, remuneration rule has prevented this scenario to happen, let's assume there is a fragment and DS which only has G number of copies in the network ($G = 2$ in this experiment), the changes of possibility of losing this fragment is showed in Figure.24., the highest probability is only tiny above 6%.

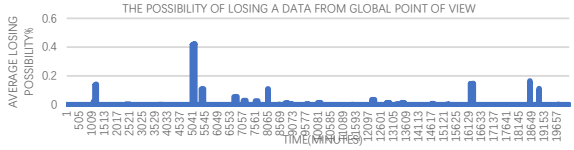

Fig. 23. The possibility of losing a data globally

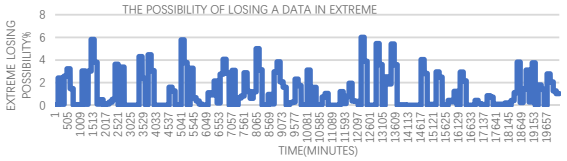

Fig. 24. The possibility of losing a data in extreme

5.2 *Speed test and Byzantine identification*

To prevent the Byzantine block publisher from receiving remuneration for the fragment and DS it pretended to hold. The protocol ruled that the Byzantine identification and Speed test cannot use a φ which points to a block or DS that can disclose parts of information of the Merkle branch which is legal to write into the Proof of Storage Box at current block height of any known forks. If it sends this kind of PoS, its peers will not retransmit the enquiry for it. And there is also

restriction for selecting PoS parameters, which are distributed in section 3.5.1.

5.3 Jump distance

Byzantine identification method has prevented the possibility of sham paths. However, it cannot prevent a misreport of jump distance. If a Byzantine node falsely reports extreme jump distances of a fragment to its peers, and if the network distances paired to these jump distances happened to be among the shortest, this false information might make nodes grab/abandon specific fragment and DS. However, after adjustment storage, when the fragment and DS become overabundance/insufficient in the network, other nodes will drop/add this fragment and DS, which will eventually make the storing balance again.

5.4 Sybil attack

A Byzantine may simulate tons of nodes in the network to conduct sybil attack, which is making specific fragments and DSs appear to be overabundant in the neighborhood. Nodes might be induced by them to drop relevant data. However, that is hard in the real world, when the nodes are around the world, the simulated nodes in a physical area lose the advantage of network distance in another. To make a node A drop specific data, the Byzantine must first have this data, and it must manage to be in A's neighborhood. To be in A's neighborhood, Byzantine must specially adjust its peer structure to achieve a high SP with A; it must continuously retransmit data that is matching to the SP to avoid A disconnect it. To prevent other nodes in A's neighborhood syncing the data from different sources after A dropped the data, Byzantine must be very close to A in network distance, and the Byzantine must be the new source of this data after A dropped the data. However, it is tough for Byzantine to induce the majority keepers of specific data to dump the data simultaneously because every node has a random storage change window. In addition, because the existence of guarantee node, the Byzantine cannot induce all the nodes to drop the data.

5.5 Getting control of Guarantee nodes

If an attacker wants to destroy a fragment and DS, it must get control of all the guarantee nodes of this fragment and DS, and it must simulate lots of nodes globally to keep this fragment and DS as to induce others to drop this fragment and DS. The chance of success is minimal. Since as a guarantee node, nodes only need to stay online and keep a specific fragment and DS and then they are guaranteed to receive remuneration after its duty time as guarantee node, nodes are motived to compete to become a guarantee node, the financial demand for the attacker is high. After system run for a long time, several guarantee nodes of different fragment and DS will expire at the same block height. Followed the rule of selection, the block publisher randomly pairs the new guarantee nodes to the fragment and DS they need to keep by the block publisher. Thus, there are several possibilities. It is true that the attacker may become guarantee nodes of the fragment and DS it targeted but it is also possible that the attacker becomes the guarantee nodes of other fragment and DS. The number of guarantee nodes needs to be replaced at block height BH is$\mathrm{Q} = \mathrm{int}\left(\frac{\mathrm{BH}}{5000}\right) \times \left(\max(0, \min\left(\mathrm{Ceil}\left(\frac{G}{T}\right), \mathrm{G} - \mathrm{Ceil}\left(\frac{G}{T}\right) \times (\mathrm{BH\ MOD\ T})\right)\right)$, provided there is no guarantee node violated its duty and being replaced earlier. Supported an attacker wants to target a fragment and DS and there are $\mathrm{Ceil}\left(\frac{G}{T}\right)$ number of guarantee nodes of this fragment and DS needed to be replace at this block height; if the attacker has successfully sent U number out of the Q most significant amount of frozen transactions that are written in this block; the chance for the attacker to become all the $\mathrm{Ceil}\left(\frac{G}{T}\right)$ number of the guaranteed nodes of the fragment and DS it targeted is $\mathrm{P} := \frac{\binom{U}{\mathrm{Ceil}\left(\frac{G}{T}\right)}}{\binom{Q}{U}}$. For example, let G=2, T=4, $\mathrm{U} = \frac{Q}{2}$; at block height 50,000 $\mathrm{Q} = 10$, if $\mathrm{U} = 5$, $\mathrm{P}=\frac{1}{126}$; at block height 500,000 Q= 100, U= 50, $\mathrm{P}= \frac{25}{50445672272782096667406248628}$. The example shows that even if the attacker has half the amount of funding selected for a round guarantee node selection, the chance for it to be assigned to keep all the fragment and DS it targeted is minimal and is decreasing along with the aggregation of blocks. Not to mention the chosen of all the guarantee nodes of a specific fragment and DS is divided into several times of guarantee node selection.

VI. DECENTRALIZED STORAGE SYSTEMS

The merits of using section-blockchain to power decentralized distributed storage system is as follows:

(1) Autonomous global storage adjusts without any centralized control.

(2) Quick response. As nodes getting a reward for keeping data, nodes are promptly to detect any network/storage change and to adjust its local storage to achieve a win-win of local optimization for global optimization.

(3) Data are equally distributed among the network, which means users all over the world can access data quickly.

(4) The storage can be outsourced. In the old models, users treat data as authentic if they download the data from authenticated nodes. In the new model, users can use blockchain to check the genuine of a data, thus data can be stored by anyone. The boundary between user and service provider can be blurred.

(5) Robust. As the experiment showed, the section-blockchain can withstand 20% of node losing in a sudden without losing any data. This feature safeguarded the merit (4) because outsourced nodes are often unreliable and may go offline without prior notices.

(6) DDOS freed. In the old models, the servers are usually few compared to user scales. Thus, the DDOS attack for any server can damage or partly stop the servers. When the system outsources the storage to its users, the actual service provider can be numerous even a chrome extension can be a node, the DDOS attack cannot easily damage the system.

(7) Data protection and privacy strengthen. Data in section-blockchain travels around the network without an easy predicted path, the analysis we gave in section 5 also suggests it is impossible to destroy a data maliciously. Besides, it reinforced the privacy of the Internet because it is hard to tell the purpose of retrieving data (download file or a local storage change).

It is easy to expend Section-blockchain to decentralized storage by placing a data storage mechanism simulated with the storage of fragment and DS introduced in above sections. Every node additionally maintains a file grabbing table and several parts of files. We can also implement the reward from keeping data. Users only need to split their files into pieces and inform the network with the Merkle Roots of those file, just like report transactions into Bitcoin. The content of data is not written into blocks but only the Merkle Roots of the file pieces. Users then send all the parts of data to peers; a global backup will trigger. If a user wants to delete its file, it informs the network using a command of erasing; when a block

embedded this command, nodes globally will remove this file fragments. There will be no way to eliminate a file without the owner's permission. Data will exist forever or to the end of its rent period if the protocol is asking a renting fee for storage space of every file. If the renting fee goes back to the network, this provided the financial liquidity of this cryptocurrency.

VII. Related work

Many decentralized storage systems based on Blockchain is purposed [16] [18-19]. However, the research toward reducing the storage of Blockchain itself is still rare to see, many IoT approaches [12-15] are giving nodes different responsibility, and the nodes are not equal. To the author's knowledge, there is not research toward cutting the storage of blockchain while maintaining the equality of all the nodes is purposed before this paper. Many existing blockchains powered decentralized storage system never assign files in the way of equal distribution globally as like section-blockchain did, they, in fact, are just using blockchain as the trading system between the file owners and file keepers. Though the files are multi-backup, the positions of the nodes which kept the data are arbitrary in the network. Thus, not like section-blockchain, the speed for retrieving data is also verified globally. Also, without equal distribution globally, it goes without saying that these systems are more vulnerable when losing nodes of massive scale. Among many systems, the most popular one is IPFS. IPFS represents the InterPlanetary File System, which is a peer-to-peer distributed file system, aims to replace HTTP. IPFS synthesizes many of the best ideas from the most successful systems to date [17]. Compared to our communication protocol, it uses BitSwap to enhance to the efficiency of the network. BitSwap Credit is a simple credit-based system which solves the problem of the unbalance between work and benefit. The debt ratio becomes a measure of trust which incentivize nodes to exchange lots of data.

Moreover, not like our communication protocol which is completely decentralized, BitSwap Ledger is critical to a connection between BitSwap peers. BitSwap peers are looking to acquire a set of blocks (want_list) and have another set of blocks to offer in exchange (have_list) [17].

However, IPFS's Object Merkle DAG is an excellent part of the system because Content Addressing, Tamper resistance, and Deduplication are all using properties from that design. This design can be combined with the advantage of section-blockchain.

VIII. Conclusion

Section-Blockchain allows users to store parts of blockchain but gain the full security as like full nodes in Nakamoto Blockchain. Section Blockchain also secured the system from excessive lightweight nodes and allowing users to join in the system not only to contribute calculation power but also to contribute storage. It is possible for a decentralized, autotrophic Storage System to be built above section blockchain along with a digital currency of real value anchor in it. Comparing to the existing decentralized system, the data is more distributed and withdrawn sudden dropping of nodes.

However, the whole idea of section-blockchain based is nodes are driven by financial interests, if the mining is trending to be centralized, there will not be that many nodes in the network to contribute their storage. Thus, in the future research, how to eliminate the mining power centralization would be the key to further secure section-blockchain.

In addition, the bottom line of storage, as well as the guarantee nodes needed to be selected out per block, will go up time by time after the system reached a steady user scale, this will decrease the number of transactions a block can handle since the block will need to include so many PoS enquires and guarantee transactions. The future research may also focus on how to reduce the block size, for example, using IBLT and Bloom filter to create lightweight blocks etc.

References

[1] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/ bitcoin.pdf, 2008

[2] Elbahrawy A, Alessandretti L, Kandler A, et al. Bitcoin is Not Alone: Quantifying and Modelling the Long-Term Dynamics of the Cryptocurrency Market[J]. Social Science Electronic Publishing, 2017.

[3] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S. A fistful of bitcoins: characterizing payments among men with no names. In Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013 (2013), pp. 127–140.

[4] CoinDesk. Bitcoin venture capital. http://www.coindesk.com/bitcoin-venture-capital/, retrieved Sep. 2017.

[5] Marketwatch. Bitcoin rises again setting another record http://www.marketwatch.com/story/bitcoin-rises-again-setting-another-record-2017-08-31, retrieved Sep. 2017.

[6] Bitinfochart,total storage demands. https://bitinfocharts.com/bitcoin/, retrieved Sep. 2017

[7] Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. Cryptology ePrint Archive, Report 2015/675, 2015. http://eprint.iacr.org/.

[8] The Ethereum community. Ethereum white paper. https://github.com/ethereum/wiki/wiki/ White-Paper, retrieved July. 2015.

[9] Bitcoin, developer-guide. https://bitcoin.org/en/developer-guide#simplified-payment-verification-spv, retrieved Sep 2017.

[10] Investopedia, segwit-witness. https://www.investopedia.com/terms/s/segwit-segregated-witness.asp, retrieved Sep 2017.

[11] Yibin X. An Efficient Communication Protocol for Distributed Ledger Technology. http://gofun.online/document/communication-protocol.pdf, Oct 2017. (in press)

[12] A. Dorri, S. S. Kanhere, R. Jurdak, P. Gauravaram, Lsb: A lightweight scalable blockchain for iot security and privacy, arXiv preprint arXiv:1712.02969.

[13] N. Apthorpe, D. Reisman, N. Feamster, A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic, arXiv preprint arXiv:1705.06805. [10] R. C. Merkle, A digital signature based on a conventional encryption function, in: Conference on the Theory and Application of Cryptographic Techniques, Springer, 1987, pp. 369–378.

[14] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper 151.

[15] C. Cachin, Architecture of the hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016. [13] S. Huh, S. Cho, S. Kim, Managing iot devices using blockchain platform, in: Advanced Communication Technology (ICACT), 2017 19th International Conference on, IEEE, 2017, pp. 464–467.

[16] IPFS, https://ipfs.io, [Online; accessed Jul. 28th, 2017].

[17] Benet J. Ipfs-content addressed, versioned, p2p file system[J]. arXiv preprint arXiv:1407.3561, 2014.

[18] Ali M, Nelson J C, Shea R, et al. Blockstack: A Global Naming and Storage System Secured by Blockchains[C]//USENIX Annual Technical Conference. 2016: 181-194.

[19] Tamo I, Wang Z, Bruck J. Zigzag Codes: MDS Array Codes With Optimal Rebuilding[J]. IEEE Transactions on Information Theory, 2013, 59(3):1597-1616.

[20] Josefsson, S.; Liusvaara, I. (January 2017). Edwards-Curve Digital Signature Algorithm (EdDSA). Internet Engineering Task Force. doi:10.17487/RFC8032. ISSN 2070-1721. RFC 8032. Retrieved 2017-07-31.