Technical description of the implementation of a distributed file storage based on the Luxcore blockchain

Contents

Stages of development

Description	**Hours**
Implementation of the block file storage	80
Implementation of the node interaction protocol for the functioning of dfs (decentralized file system)	160
Implementation of dfs RPC commands	40
Implementation of the generation and verification algorithm of Merkle path to prove file storage	40
Implementation of the reward system for file storage (transactions)	80
Testing	40
Total	440

Description of various flows

Description of the general principle of the entire network

PMN organize a network among themselves, by analogy with the usual masternodes (but the functionality of PMN-nodes will be completely different from the masternodes functionality)

Each member of this network provides the same fixed size of their disk space that will be used by dfs.

A user of any lux network node may request to add multiple copies of his file(s) to the PMN network for a time-based fee. For this, user needs to create a transaction of a certain type that stores the ip-address of the node that created the request, which will not be added to the blockchain but will be transmitted over the network in the same way as normal transactions.

Each PMN-node receiving the request checks its remaining free space (X bytes) and sends its proposal for storing the file to the node that created the request.

The node that initiated the request, receiving a proposal to store a file from a PMN node can verify the accuracy of the received data about the free space of the PMN-node, having information in its blockchain about how much data PMN-node currently stores.

A node that sent a request to add a file waits for a predetermined time (network parameter) receiving messages from all PMN-nodes ready to fulfil its order and sorting them by priority (free space on the PMN node). Then selects n PMN-nodes, where n is the parameter specified in the original transaction as the number of copies of the file. And sends a message to these PMN-nodes for file transfer.

Next the file(s) are transferred to all selected PMN-nodes with a record of the Merkle root file(s), the size of the file(s) on the blockchain.

The PMN node storing the file has the right once every *t* minutes (where *t* should be much longer than block mining time) to make a transaction, with proof of file storage and receive a reward equal to *p* * *t* * *s*, where *p* is the rate for storage (network parameter), *s* - size of stored data.

The user of any node of the lux network node can download the file saved in dfs, specifying the Merkle root of the desired file.

The user of any node of the lux network node can check a file saved in dfs, indicating Merkle root the desired file.

Description of user functions

- Count the Merkle root file or group of files on your local machine (parameters: path to the file or folder)

- Polling of PMN-nodes, in order to find PMN-nodes that can save the file in dfs (parameters: Merkle root file(s), the timestamp to which the file will be stored, the address from which funds will be written off)

- Transferring a file from Alice to Bob, with the creation of StorageOrderTx by Alice. (parameters: the path to the file to be saved in dfs on the local machine)

- Check file availability in dfs (parameters: Merkle root file(s))

- Download file from dfs (parameters: Merkle root file(s))

Description of the PMN-node interaction protocol with dfs

Description of block file storage

A group of files to be saved is recorded in memory sequentially and working with it is equivalent to working with a single file.

The file is divided into blocks of the same size (size is a parameter). The last block is complemented by “nulls” to the desired size. For each block a hash is calculated using the number of bytes actually stored in the block.

For each block the sha-256 hash function is considered, the Merkle tree is formed from these hashes, the root of which is subsequently stored in the blockchain.

Description of the file storage proof algorithm

The PMN-node keeps a file and a full Merkle root for it.

Once in *t* minutes, a node can generate a transaction to receive a reward.

In this transaction, k evidence is stored (network parameter), each of which corresponds to a deterministic Merkle tree path, depending on the previous path and iteration number i = 1..k i.e. in each proof transaction the PMN-node provides k blocks of the stored file, their hashes and the corresponding Merkle paths. The choice of k blocks for each proof case is determined by the corresponding blockchain algorithm and depends on the previous proof.

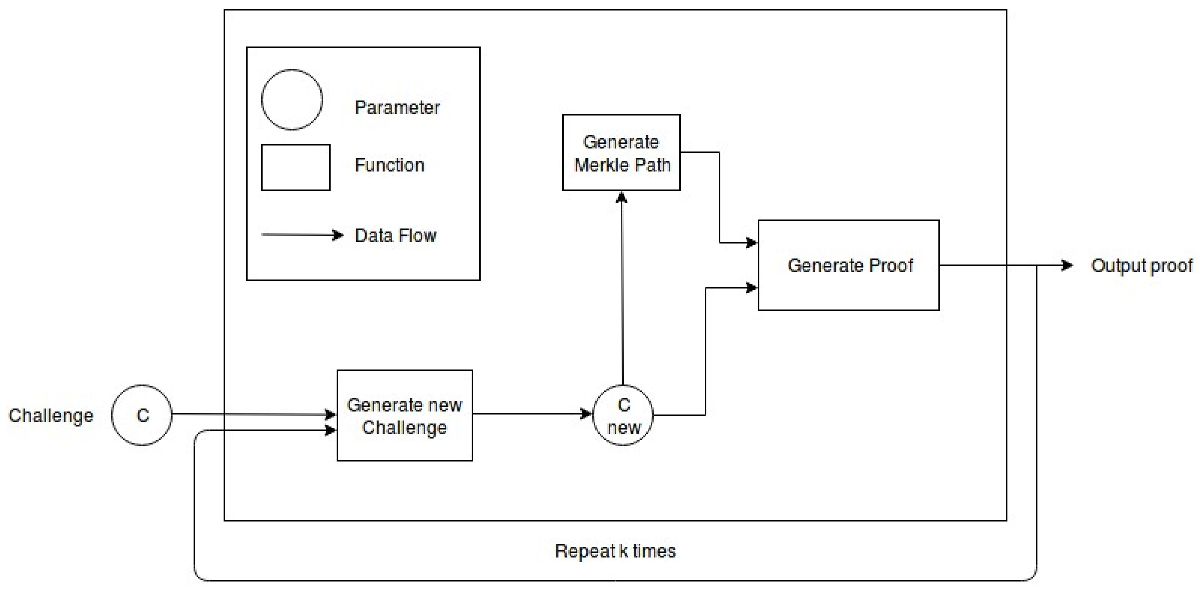

Fig 1. The scheme of the file storage proof algorithm

Description of the reward system for file storage

When executing a request to save a file a DfsSubscribeTx transaction is created, which at the user’s new address blocks the funds equal to *p* * *t* * *s*, where *p* is the storage rate (network parameter), *t* is the entire desired file storage time, *s* is the file size. In the future of these funds will be issued a reward to PMN-node, which stores the file of this user. The above process is applied asynchronously for each copy of the stored file (for each PMN-node, a separate transaction Dfsr SubscribeR is created by the initiator).

Each transaction with a file storage proof created by a PMN-node, uses the output of the address of the corresponding DfsSubscribeTx transaction, sending the “change” back to the same address. Thus, from the reserved funds, file storage payment is made periodically, during the specified storage period.

The initiator can extend the storage period for the file by creating a new DfsSubscribeTx transaction for the same file and the same PMN-node that stores this file, and specifying the new retention period.

When the file has expired, the PMN-node ceases to guarantee that the file is stored for the rest of the network.

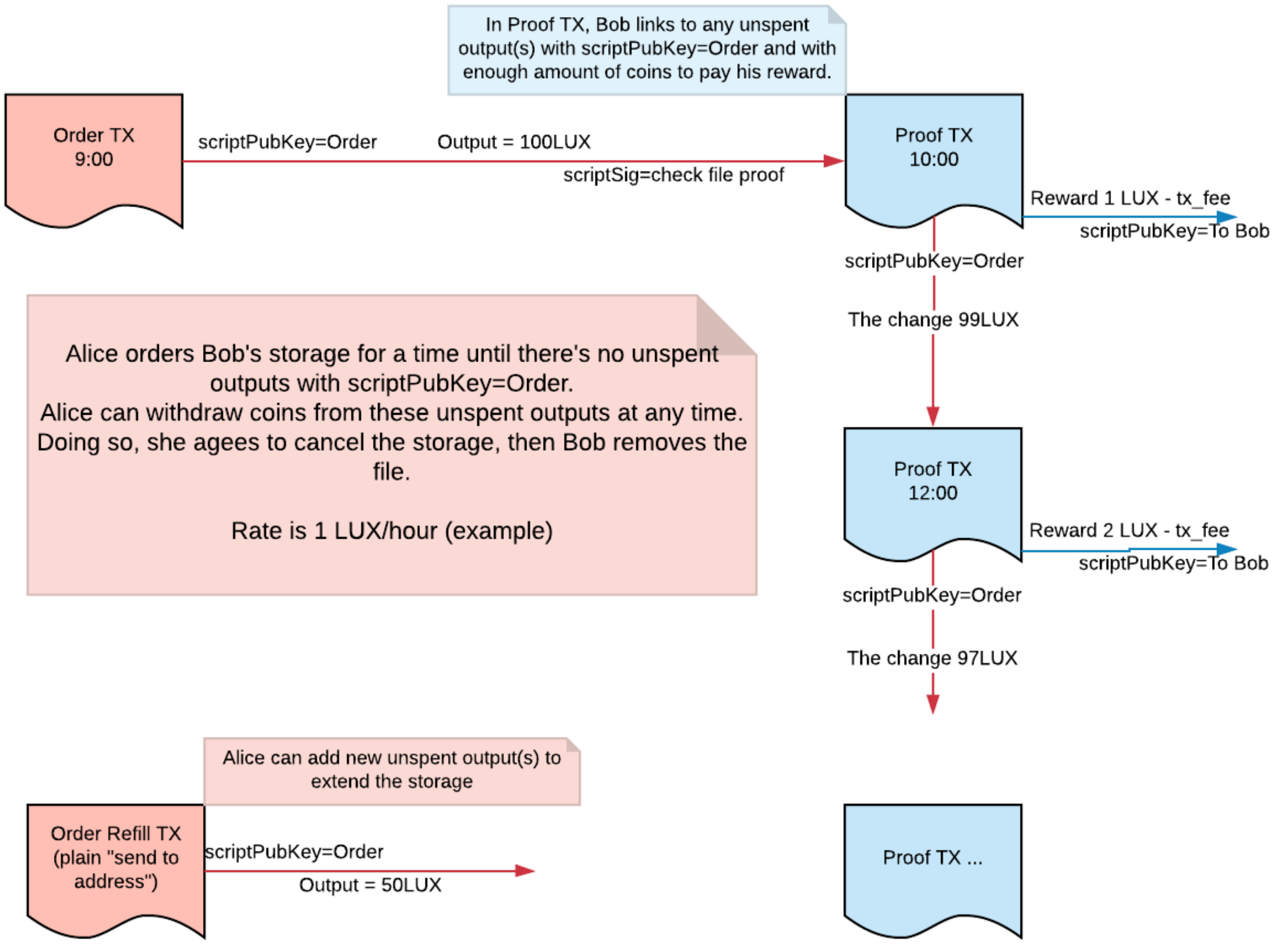
In Proof TX, Bob links to any unspent output(s) with scriptPubKey=Order and with enough amount of coins to pay his reward.
Order TX 9:00
scriptPubKey=Order
Output = 100LUX
scriptSig=check file proof
Proof TX 10:00
Reward 1 LUX - tx_fee
scriptPubKey=To Bob
scriptPubKey=Order
The change 99LUX
Alice orders Bob's storage for a time until there's no unspent outputs with scriptPubKey=Order.
Alice can withdraw coins from these unspent outputs at any time.
Doing so, she agees to cancel the storage, then Bob removes the file.
Rate is 1 LUX/hour (example)
Proof TX 12:00
Reward 2 LUX - tx_fee
scriptPubKey=To Bob
scriptPubKey=Order
The change 97LUX
Alice can add new unspent output(s) to extend the storage
Order Refill TX (plain "send to address")
scriptPubKey=Order
Output = 50LUX
Proof TX ...

Fig 2. Scheme of the PMN-node reward system for file storage.