Split-Scale: Scaling Bitcoin by Partitioning the UTXO Space

Preprint · September 2018

CITATIONS
0

READS
5

3 authors, including:

Kazım Rıfat Özyılmaz
Bogazici University
7 PUBLICATIONS **9** CITATIONS

SEE PROFILE

Accepted for publication in 9th IEEE International Conference on Software Engineering and Service Science (ICSESS 2018) on 09.07.2018 - published version may differ

Split-Scale: Scaling Bitcoin by Partitioning the UTXO Space

Kazım Rıfat Özyılmaz
Department of Computer Engineering
Bogazici University
Istanbul, Turkey
kazim@monolytic.com

Harsh Patel
Ahmedabad, India
reach@harshpatel.space

Ankit Malik
New Delhi, India
ankit@ankitmalik.in

***Abstract*—The Bitcoin protocol is a significant milestone in the history of money. However, its adoption is currently constrained by the transaction limits of the system. As the chief problem of blockchain technology, the scaling issue has attracted many valuable solutions both on-chain and off-chain.**

In this paper, our goal is to explore the notion of unspent transaction outputs (UTXOs) to propose an augmented Bitcoin protocol that can scale gracefully. Our proposal aims to increase the transaction throughput by partitioning the UTXO space and splitting the blockchain. In addition, a new type of Bitcoin node is introduced to preserve the capability to run validating nodes in low-bandwidth environments, despite the increased transaction throughput.

***Keywords*—Bitcoin, blockchain, consensus, UTXO, scaling, block size**

I. Introduction

Scalability is the one of the most important aspects affecting Bitcoin's adoption. Limits of scalability express themselves as high transaction fees, which affects usability and adoption negatively. To improve transaction throughput, various proposals have been made, starting with directly increasing block size. However, the most effective scaling improvement already integrated to Bitcoin is 'Segregated Witness' [1]. This increased the block capacity by introducing the block weight metric. Another notable attempt to solve the scalability problem is the Bitcoin-NG protocol [2] which introduced an additional mining process where miners gain the capability to mine microblocks by mining a regular Bitcoin block.

In this paper, we propose a solution that increases the transaction throughput of the Bitcoin network without hurting network decentralization in terms of bandwidth requirements. By partitioning the UTXO space and splitting the blockchain into a tree structure, independently operating sub-chains will be created at every split event. As a result, a new block from all sub-chains will be mined at every block interval, increasing the transaction throughput exponentially. Moreover, in order to preserve the capability to run a node in this increasing bandwidth requirement, a new type of Bitcoin node (the half node) is introduced. Although this new node type does not store the complete blockchain, it can independently verify the transactions on the sub-chain it is tracking, which gives it the capability to operate in low-bandwidth environments.

In the next section (Section II), an overview of the core concepts is presented. Then, the general mechanics and technical details (Section III) of the proposal are described. Effects of the proposal to mining (Section IV) and network organization (Section V) are discussed in the following sections. Next, the transactions discussion (Section VI) provides insights on transactions in the proposed system. A section dedicated to comparison of the split-scale proposal to other major Bitcoin scaling solutions (Section VII) is included afterwards.

II. Core Concepts

A. Unspent Transaction Output (UTXO)

Bitcoin does not use the concept of 'account balance' as Ethereum does. Instead, total balance of a Bitcoin account is the accumulated amount of the *transaction outputs* that are claimable but not yet spent. These *unspent transaction outputs* or *UTXOs* for short, are used as the inputs of the transactions. They are referred using the *source transaction hash* and *index of the output* within that source transaction (Listing 1).

```
// transaction input
class CTxIn
{
public:
    COutPoint prevout;   // UTXO-to-spend
    CScript scriptSig;   // input script
    uint32_t nSequence;
    CScriptWitness scriptWitness;
};
// pointer to transaction output
class COutPoint
{
public:
    uint256 hash;        // transaction hash
    uint32_t n;          // index of the output
};
// transaction output
class CTxOut
{
public:
    CAmount nValue;       // amount of Bitcoin
    CScript scriptPubKey; // output script
};
```

Listing 1: Transaction Input [3], OutPointer [4] and Output [5]

The receiving party will have at least one transaction output after the transaction is validated and added to the blockchain. If a UTXO is used as a transaction input and the transaction

is already a part of the blockchain, then it is considered spent and thus can not be used a second time as a transaction input.

The UTXO set is stored by nodes in a database called *chainstate.db* outside the blockchain, which provides persistent key-value storage. As of Bitcoin 0.15.0, the chainstate database has been changed from a per-transaction model to a per-output model which added benefits like faster serialization, predictable memory usage and better caching [6]. This change may provide a smooth transition for the chain splitting mechanism that is proposed in this paper.

The *CTxIn* class (Listing 1, line 2-9) is a simplified version of the transaction input. It contains the location of the previous transaction's output that it claims and a signature that matches the output's public key. The *COutPoint* class (Listing 1, line 11-16) in the transaction input shows how UTXOs are actually referred by the input. It contains both the *transaction hash* and the *index* of its output. Lastly, the *CTxOut* class (Listing 1, line 18-23) presents the anatomy of a transaction output in a simplified form. It contains the amount and the script *scriptPubKey* to claim the output. Below is the *scriptPubKey* for a standard Pay-to-PubkeyHash (P2PKH) transaction [7]:

```
OP_DUP OP_HASH160 pubKeyHash OP_EQUALVERIFY OP_CHECKSIG
```

On a side note, as of 26th of October 2017, 82% of the Bitcoin transactions are Pay-to-PubkeyHash (P2PKH) [8] so *scriptPubKey* of these transactions are directly tied to a single receiving address.

B. Memory Pool (mempool)

The *memory pool*, or *mempool*, is the memory area reserved by Bitcoin clients to store unconfirmed transactions. Unconfirmed transactions are accumulated in the mempool until they are picked by a miner, mined, and added to the blockchain. Currently, each node maintains its own mempool, having the complete view of all unconfirmed transactions in the Bitcoin network. The amount of memory reserved for mempool varies greatly, the peak point being around 140MB for the last two years [9].

III. Split-Scale

The idea is to split the Bitcoin blockchain (Figure 1), known here as *split events*, into multiple sub-chains in order to:

- create independently operating multiple sub-chains, therefore creating multiple blocks instead of one block for every block creation interval (an interval lasts 10 minutes on average).
- provide the flexibility of operating home nodes with less bandwidth and storage requirements. These nodes will have the option to track only a subset of chains without losing any verification capability.

The mechanics of such a split and how UTXO database, mempool or mining operations will be affected will be presented in the following subsections.

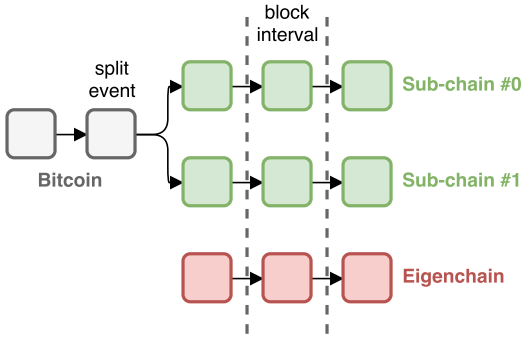

Fig. 1. Bitcoin after Split Event

A. Split Event

A *split event* is a deterministic (and repeatable) action that will be triggered as a result of a decision made by the governing authority of the platform and/or certain performance metrics showing that the system is pushing its boundaries in terms of transaction throughput. Details on when a split event will occur or who is going to decide for it is a separate topic that will not be addressed in the scope of this paper. At the split event, following changes will happen on Bitcoin clients:

- UTXO database will be divided based on their *scriptPubKey* hashes.
- the UTXO split may be implemented logically (UTXO hashes in binary form that start with 0 or 1 for the first split) or economically (find the 256bit number that divides the Bitcoin supply in half).
- the mempool will be divided based on which sub-chain they are tracking.
- miners must create a block for each sub-chain, and a separate block containing these block headers to claim a block reward.

B. Dividing the Chainstate Database

Based on the information given on the UTXO structure, it is possible to create a 256-bit hash value from the *scriptPubKey* of any given UTXO. The hash function is selected as double SHA256() and a hash value representing all UTXOs with the same *scriptPubKey* can be given as follows:

$$\text{hash}_{\text{UTXO}} = \text{SHA256}(\text{SHA256}(\text{CTxOut.scriptPubKey}));$$

This approach will work for all the script types including Pay-to-PubkeyHash (P2PKH), Pay-to-ScriptHash (P2SH), Pay-to-multisig (P2MS) or Pay-to-Pubkey (P2PK) outputs.

At the split event, the Bitcoin client will calculate 256-bit hash values for all UTXOs and decide in which chainstate database a UTXO will end up. At that point, the Bitcoin client will create *chainstate0* and *chainstate1* databases and remove the original *chainstate.db*, in case of a two-way split.

From the split event onward, sub-chains start to add their own blocks. After the first valid blocks are added, two sub-chains behave just like mini-Bitcoin networks independent of each other.

C. Dividing Mempool

At the split event, Bitcoin memory pool will flush, create *mempool0* and *mempool1* and remove the original mempool. Starting with the next block, only transactions that belong to a specific sub-chain will be considered valid. Each sub-chain will transmit its unconfirmed transactions to a different mempool, and they will be picked up by miners separately for each sub-chain.

D. After the Split

Dividing a UTXO set and mempool will create mini-Bitcoins effectively (Figure 1). These mini-Bitcoins will function independently by conforming the rules below:

- every full node can divide UTXO set and mempool, and keep track of the divided sets independently based on the guidelines (as additional consensus parameters coded in the Bitcoin client).
- mempools will detect (using the new consensus parameters) and won't accept mixed transactions, e.g. users can't mix up UTXOs belonging to different sub-chains in a single transaction.
- the UTXO set will be tracked in multiple databases, on a per-subchain basis.

A side effect of this is the continued presence of regular Bitcoin addresses in all sub-chains. However, there may be a different number of UTXOs attached to those accounts on every sub-chain. The effects of such an event on mining and network organization will be elaborated on in the following sections.

IV. Mining

Bitcoin uses Proof-of-Work (PoW) consensus that utilizes double SHA256() as its hash function. Every new block should contain the hash of the previous block in its header and the checksum of its own header should be lower than the 256-bit difficulty value that is updated every week. The proposed scaling solution will not attempt to change the hash function or propose a new consensus function. The aim of this paper is to adapt the current mining approach to a multi-chain setup.

As described in previous section, after the split event there are multiple sub-chains that can operate independently. This means every sub-chain may separately mine their own blocks and append to the blockchain, assuming all mining parties are honest. However, to keep the Bitcoin system robust and trustless in a multi-chain setup, some form of super-mining should be enforced, instead of making assumptions about the honesty of the other parties. Otherwise, miners with great computing power will jump through sub-chains depending on the difficulty values, which will make the system more vulnerable to instantaneous attacks [10].

A. Eigenchain

In order to keep to mining power in check, adding a new block should happen atomically across on all sub-chains. This means the block count will be the same across all sub-chains, all the time. However, to verify the newly added blocks and detect double-spend attacks, there should be a separate blockchain that keeps track of the all the blocks added to their respective sub-chains. This new blockchain that stores the block header hashes of sub-chain blocks is called *eigenchain*.

The mining will works as follows:

1) Miners have to listen to all sub-chains and pick transactions from all mempools.
2) Miners have to mine a block for every sub-chain (the difficulty levels will be adjusted after the split events).
3) Miners have to create a new eigenchain block by using sub-chain block headers. The difficulty of the eigenchain block will be higher than the sub-chain blocks, almost constituting half of the difficulty in Bitcoin network at the time of the split event.

In Bitcoin, a newly mined block is serialized [11] and transmitted using the *block* message [12]. In this proposal, however, the *block* message will transmit a single serialized block similar to Bitcoin but that block will contain both the new eigenchain block and all the other sub-chain blocks.

Although the proposed approach seems like a glorified block size increase at the moment, the changes in the network organization and introduction of the *half node* will show the benefits of the approach. For an explicit comparison, refer to the Section VII: "Bitcoin Proposal Comparison." The network organization will be discussed in the following section.

V. Network

The Bitcoin network consists of multiple types of peers: miners, full nodes and lightweight nodes. Miners are the peers that create and transmit new blocks to the network, full nodes are the verifiers that store the complete blockchain and lightweight nodes are the relatively weak ones that use *Simple Payment Verification (SPV)* to only verify particular transactions [14].

A. Full Nodes

In the regular Bitcoin network, full nodes store the complete blockchain and execute block and transaction verifications all the time to keep the system secure. Similarly, in our proposal, full nodes will keep in sync with all sub-chains plus the eigenchain, therefore it will be able to verify a specific sub-chain in itself and cross-reference it with the eigenchain. Miners and full nodes are connected in a way similar to the current Bitcoin network formation and new *block* messages are only sent to full nodes. Full nodes will verify and update the newly mined blocks, and will then re-transmit the sub-chain blocks (a serialized eigenchain block and appended sub-chain block) to the relevant networks formed by sub-chain nodes. In short, full nodes are interconnected to full nodes and half nodes. Not all messages are sent to sub-chain networks, however. Only the relevant ones are propagated to minimize the bandwidth requirements.

B. Half Nodes

With the proposed scheme, an additional type of node called *half node* is added to the system. Half nodes keep track of one

TABLE I
SPLIT-SCALE COMPARISON: MINER PERSPECTIVE

Miner Features	Bitcoin (SegWit)	SegWit2x	Bitcoin-NG	Split-Scale
Scale Factor	1x	2x	60x	Nx (scales exponentially with split count)
Block Count	mine one block	mine one block	mine one key block plus microblocks (every 10s)	mine one block on all sub-chains plus one eigenchain block
Block Size	~1MB on average 4MB SegWit limit	~2MB on average 8MB SegWit limit	same as Bitcoin (SegWit)	same as Bitcoin (SegWit)
Transaction Fees	from one block	from one block	from all key and microblocks	from all sub-chain blocks

TABLE II
SPLIT-SCALE COMPARISON: NODE PERSPECTIVE

Node Requirements	Bitcoin (SegWit)	SegWit2x	Bitcoin-NG	Split-Scale
Storage Requirements	whole blockchain	whole blockchain	whole blockchain	full nodes store the whole blockchain half nodes store only one sub-chain and eigenchain
Bandwidth Requirements	at least ~700Kb [13]	at least ~1.4Mb	full node bandwidth requirements increase linearly with scaling factor (60x)	full node bandwidth requirements increase linearly with scaling factor (Nx) half node requirements will be similar to Bitcoin (SegWit)

sub-chain and the eigenchain. A half node is able to verify both the tracked sub-chain and eigenchain blocks by using block header hashes and is able to cross-check and validate sub-chain blocks using information contained in eigenchain. Half nodes only keep track of one mempool and one chainstate database (UTXO set) depending on which sub-chain they select. In addition, half nodes do not get new *block* messages. New blocks targeting sub-chains are transmitted by using a new type of message: a *block-n* message, which contains only the serialized eigenchain and sub-chain block. This way both the storage and bandwidth requirements of half node will be significantly lower compared to full nodes.

VI. TRANSACTIONS

After the split event, all the UTXOs of a specific *scriptPubKey* will be accumulated in one sub-chain. Basically, users will be able to create transactions using only UTXOs from a specific sub-chain and be able to transact without knowing the remaining sub-chains. However, in time, users will receive payments from multiple parties in different sub-chains. Therefore total account balance of a user will more or less reside in multiple sub-chains with different UTXOs attached to it. If a user wants to spend more than the total amount of Bitcoin in one of his sub-chains then multiple transactions should be made.

A. Hashed Time-Lock Contract (HTLC)

Hashed Time-Lock Contract, or *HTLC* in short, is defined as: 'a class of payments that use hashlocks and timelocks to require that the receiver of a payment either acknowledge receiving the payment prior to a deadline by generating cryptographic proof of payment or forfeit the ability to claim the payment, returning it to the payer' [15]. Lightning Networks use HTLC to be able to construct secure transfers using a network of channels across multiple hops to the final destination [16].

In the proposed system, HTLCs are used to ensure the atomicity of the payment, even the payment consists of multiple transactions on multiple sub-chains. Assuming that the sender does not have enough balance on one sub-chain to cover the complete payment, then the sender has to create multiple transactions on multiple sub-chains respectively. The receiver may claim each transaction on a different sub-chain, but it is preferred to finalize the payment in a single step. In such cases, senders (therefore the underlying Bitcoin wallet implementations) should utilize HTLCs to ensure atomicity. The sending process should be as follows:

1) the sender creates random data.
2) the hash of that random data is calculated.
3) the hash value is added to all transactions (*scriptPubKey*) and transactions are sent on their respective sub-chains.
4) after all the transactions are mined, the total payment amount may be claimed by the receiver, after complete random data is shared by the sender.
5) if any of the transactions fails in a predefined time interval, funds may be claimed by the sender again.

B. Eigentransactions

An eigentransaction is a failsafe mechanism which may be added to the system to make fund transfer possible between the sub-chains. However, these transactions are special and limited to sending funds only between the same addresses in multiple sub-chains, so the private key of sending and claiming address should be the same. This is to provide an easy way for transferring the total account balance into a single sub-chain.

Eigentransactions should have a separate global pool called the *eigenpool* similar to the current Bitcoin mempool, and eigentransactions are mined and included into the eigenchain. This way all sub-chains will be able to track fund transfers of the same account across sub-chains and will be able to add the UTXO (if sent to that specific sub-chain) to their balance.

With the addition of eigentransactions, the block size of the eigenchain will be increased. However, activation of this feature can be easily controlled, even enabled/disabled between certain block numbers.

VII. Bitcoin Proposal Comparison

Two forms of decentralization are at the heart of the Bitcoin scaling debate. The first form is mining decentralization, which is the problem of accumulation of high hash rates at the hands of a limited number of mining cartels. The second one is decentralization of the network, which is the decreasing amount of full nodes due the increasing bandwidth requirements. Our proposal aims to scale the Bitcoin network without decreasing network decentralization. In Table I and Table II, the split-scale proposal is compared to the other valuable on-chain scaling proposals in terms of miners and network node features.

Split-scale provides a framework for scaling and gives the opportunity to scale exponentially with every split event. For the miner, our proposal will provide better economic incentives, because although the block reward is the same, the transaction fees will be collected from all sub-chain blocks. As a result, transaction fee gains for miners will even surpass Bitcoin-NG at the sixth split event (64 sub-chains) (Table I).

Finally, our solution is clearly efficient in terms of bandwidth and storage requirements (Table II). In all the other proposals transaction throughput increase (scaling) is directly translated to bandwidth and storage increase for Bitcoin nodes. As a result, due to the increasing requirements, the number of Bitcoin full nodes will decrease and network decentralization will suffer in all the other proposals. Split-scale introduces a new kind of Bitcoin node which is called 'half-node' that eliminates these restrictions and provides capability to run a node tracking only one sub-chain and eigenchain.

VIII. Conclusion

Scalability is important for expanding adoption of Bitcoin. In this paper, we address the scalability problem by partitioning the UTXO space, therefore splitting the Bitcoin blockchain into multiple sub-chains. Our approach facilitates a block creation increase due to the mining taking place on all sub-chains and it proposes a way to still maintain nodes operating in low-bandwidth conditions. Compared with prominent Bitcoin scaling proposals, "split-scale" offers scalability while preserving network decentralization.

References

[1] Bitcoin. (2015) Segregated Witness. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[2] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol." in *NSDI*, 2016, pp. 45–59.

[3] Bitcoin. (2017) CTxIn class. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/7b57bc998f334775b50ebc8ca5e78ca728db4c58/src/primitives/transaction.h#L61

[4] Bitcoin. (2017) COutPoint class. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/7b57bc998f334775b50ebc8ca5e78ca728db4c58/src/primitives/transaction.h#L18

[5] Bitcoin. (2017) CTxOut class. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/7b57bc998f334775b50ebc8ca5e78ca728db4c58/src/primitives/transaction.h#L131

[6] Bitcoin. (2017) PR #10195: Switch chainstate db and cache to per-txout model. [Online]. Available: https://github.com/bitcoin/bitcoin/pull/10195

[7] Bitcoin. (2017) Script. [Online]. Available: https://en.bitcoin.it/wiki/Script

[8] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartı, "Analysis of the bitcoin utxo set." [Online]. Available: https://eprint.iacr.org/2017/1095.pdf

[9] Blockchain.info. (2017) Mempool Size. [Online]. Available: https://blockchain.info/charts/mempool-size?timespan=1year

[10] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.

[11] Bitcoin. (2017) Serialized blocks. [Online]. Available: https://bitcoin.org/en/developer-reference#serialized-blocks

[12] Bitcoin. (2017) Block message. [Online]. Available: https://bitcoin.org/en/developer-reference#block

[13] Bitcoin. (2017) Minimum Requirements. [Online]. Available: https://bitcoin.org/en/full-node#minimum-requirements

[14] Bitcoin. (2017) Simple Payment Verification. [Online]. Available: https://bitcoin.org/en/developer-guide#simplified-payment-verification-spv

[15] Bitcoin. (2017) Hashed Timelock Contracts. [Online]. Available: https://en.bitcoin.it/wiki/Hashed_Timelock_Contracts

[16] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *draft version 0.5*, vol. 9, p. 14, 2016.