Decentralised Mining Pool for Bitcoin (Draft 0.3)

Abstract

Bitcoin P2Pool's usage has steadily declined over the years, negatively impacting bitcoin's decentralisation. The variance in earnings for miners increases with total hashrate participating in the pool, and payouts require a linearly increasing block space with the number of miners participating in the pool. We present a solution that uses a DAG of shares replicated at all miners. The DAG is then used to compute rewards for miners. Rewards are paid out using one-way payment channels by an anonymous hub communicating with the miners using Tor's hidden services. Using the payment channels construction, neither the hub nor the miners can cheat.

1 Motivation

P2Pool [1] helps bitcoin's decentralisation by allowing miners to select the transactions they mine. This avoids any potential transaction censorship by pool operators. However, the construction used by P2Pool faces a number of problems that has resulted in miners abandoning the pool. The most often cited problems are:

1. large variance in earnings for miners,
2. large number of stale blocks, and
3. large block space requirement for payouts.

The first two problems are a direct consequence of the shares block rate limited to 30 seconds on the bitcoin p2pool. Intuitively, as the hash rate participation in the pool goes up, the difficulty for 30 seconds block rate goes up and smaller miners find it harder to find shares within a 30 second period. This results in an increase in the variance for shares found by miners and thus for the rewards earned by the miners. Ethereum's inclusive protocols [5] address the problem by building a DAG of blocks and rewarding blocks that otherwise would have been left out as stale blocks. As miners get rewarded for more of their work, the variance on their earnings reduces enabling smaller miners to remain economically viable.

Payouts in P2Pool are included in the coinbase of the block being mined. As the number of participants in P2Pool increase the size of the coinbase transaction increases taking up valuable block space that the miners could have earned fees from instead. This imposes an upper bound on the number of participants in P2Pool.

Knowing the challenges faced by P2Pool, we list the goals of a new decentralised mining pool as:

1. Reduce variance for miners with increasing pool hash rate.
2. Make payouts to miners with constant size block space requirement.
3. Allow miners to select transactions they want to mine, with no potential for a censor to deny payouts.

2 Current Proposals

TeraHash Coin [6], Jute [13] and Spectre [12] use a DAG for faster block times and focus on changing the consensus layer of bitcoin. These proposals allow miners to produce shares that include conflicting transactions and then apply rules to find a set of transactions acceptable at various cuts of the DAG, such that the bitcoin consensus rules are not violated.

Braiding the blockchain proposal [6] shows how, smaller more frequent blocks can form a DAG of blocks, called a braid, with each block pointing to one or more one previous blocks. Blocks in the braid are called beads and transactions can be repeated in different beads. The proposal describes how duplicate and double spend transactions can be resolved to reach a decision on the state of the ledger at any cut of the DAG.

Belcher [2] proposes using payment channels to avoid using block space for making payouts to miners. The construction uses payment channels between federated hubs to pay miners after a block has been successfully mined. The payouts are made after a long period, similar to the 100 blocks requirements for spending from coinbase transactions. The hub locks in BTC for each miner by opening payment channels for each of the miners. The construction shows how both miners and the hub can not cheat and how the funders of the hub earns rewards for funding the payment channels.

The two ideas of using a DAG and payment channels for rewards payouts together present a potential path for rebooting P2Pool.

3 Decentralised Bitcoin Mining

In this section we describe how a DAG of miner's shares is maintained by each miner and how the same is used to determine the payout distribution between miners.

WORK

Field	Description	Size in bytes
Version	Bitcoin block's version field	4
Previous block	Hash of the previous bitcoin block	32
Difficulty	Current bitcoin difficulty	4
Coinbase	Coinbase for payment channel setup, see Section 5	38
Transactions	Transaction ids included in the block	variable
WORK hash	Hash of the above fields	32

Table 1: The structure of WORK broadcast by miners. Timestamp and nonce are left out and are included in SHAREs instead.

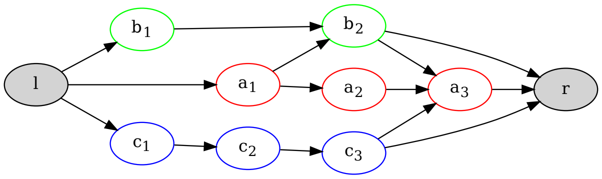

Figure 1: SHAREs mined between two bitcoin blocks

3.1 Work and Share

Each miner builds a block WORK with a structure shown in Table 1. It includes the coinbase transaction to allow other pool participants to verify the payout scripts. The transactions field is compressed using compact block specifications [3].

Miners then generate their SHAREs for this WORK and both are broadcast to the p2p network. When a miner receives a WORK from other miners it validates this WORK using a local bitcoin node. When a SHARE is received by a miner it is validated against the WORK referenced by the SHARE. Table 2 shows the structure of SHAREs broadcast by the miner.

Each SHARE includes the public keys that the hub can use to setup payment channels, and also the pubic key for the Tor hidden service run by the miner. The Tor hidden service is used for facilitating anonymous communication between the hub and the miners as described later in Section 5.9.

Share

Field	Description	Size in bytes
Work hash	Hash used to identify the work for this share	32
Nonce and extra nonce	Nonces used	$4 + 8$
Merkle root	New merkle root to include extra nonce	32
Timestamp	UNIX timestamp used	4
Public keys	Two pubkeys used by the hub to for channel management	66
list < share hashes >	List of share hashes referenced by this share	$N \times 32$
Tor Service	PubKey for onion service address, see Section 5.9.	32
Total size		$134 + N \times 32$

Table 2: The structure of shares broadcast by miners. Size of the share is dependent on the number of shares refererenced by a share (N).

3.2 DAG of shares

Miners broadcast their shares to the network using a gossip protocol and all the miners maintain the DAG of shares as a replicated database. Each node on the DAG is a share generated by a miner. Miners include hash pointers to the root nodes of the DAG, providing a "found before" relationship between shares.

Figure 1 shows a DAG with three miners participating in the pool, a, b and c. The nodes in the DAG are the shares generated by the three miners. Nodes l and r are two valid bitcoin blocks that have been mined such that they meet bitcoin's difficulty at the time.

3.3 Miner Defined Difficulty

The nodes in the DAG are shares mined at difficulty level selected by the miner. This difficulty can be dynamically changed by the miner after each share, depending on the miner's observation of the p2p network's hashrate. This dynamic difficulty adjustment allows miners to adjust the rate at which they produce shares. This is important as smaller miners can produce low difficulty shares to match the share rate of the larger miners in the pool.

Miners include the difficulty in the extra nonce they use. Our initial suggestion is that miners use 10 bits from the extra nonce field to include the multiplier they are using between the block difficulty and the share difficulty.

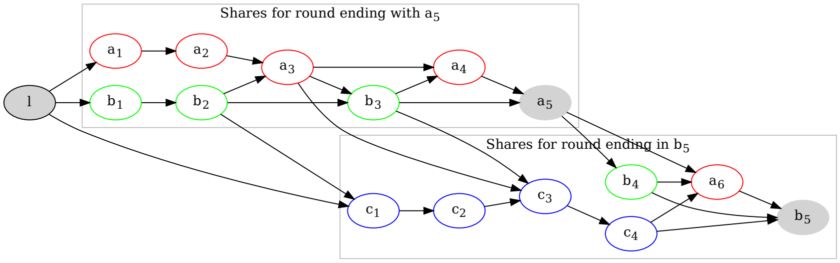

Figure 2: Reward calculation for two periods. Shares are mined by three mines — a, b and c. The shares in grey meet the bitcoin difficulty at the time they were mined.

4 Reward Calculation

We use a modified version of PPLNS (Pay Per Last N Shares) to calculate the reward distribution between miners. Variations of PPLNS have been shown to be most resistent to pool hopping [11].

1. Traverse the DAG in reverse order from the SHARE that found a bitcoin block and track the aggregate difficulty included in the SHAREs.
2. Once the aggregate difficulty reaches a system defined value, stop the DAG traversal.
3. Distribute the reward between miners weighted by the sum of the difficulty of all SHAREs found by each miner.

As an example, consider the p2p network of miners a, b and c with the DAG of shares as shown in Figure 2. In the DAG, two new blocks have been found, a_5 and b_5. For the round ending with a_5, the last "N" shares are $\{a_1..a_5, b_1..b_3\}$. The shares $c_1..c_3$ are not included in this first round as they are not reachable from a_5. However they are included in the shares for the next round ending in b_5, because their inclusion was needed to reach the total difficulty for the round. The round ending in b_5 therefore rewards the shares $\{c_1..c_4, b_4, b_5, a_6\}$.

5 Payment Channels

We propose using Payment Channels for paying miners based on the proposal by Belcher [2]. The construction of the payments channels we present is similar to Belcher's construction, but there are a few changes and we highlight them before describing the details.

Coinbase	
`2 H M 2 CHECKMULTISIG`	(cb_1)
`OR`	
`hash(X) + Hub P2WPKH`	(cb_2)
`OR`	
`M and CHECKSEQUENCEVERIFY 6 months`	(cb_3)

Table 3: Coinbase transaction with hub and miner public keys.

- In the Belcher proposal, the pool had to predict how much work miners will be able to contribute for the next block. Instead our DAG based scheme calculates rewards based on the SHAREs generated by miners.
- Belcher proposes using multiple hubs to prevent DDoS attacks on hubs, we instead use a single hub, and prevent DDoS attacks by keeping the hub hidden using Tor's v3 hidden services [4] for communication between the hub and miners.

The rest of the construction is similar to the early versions presented by Belcher — there is a one-way channel between the hub and each of the miners. The hub updates the state of each channel with appropriate reward system defined aggregate difficulty has been reached.

5.1 Coinbase

We use Belcher [2] construction where each miner builds a coinbase transaction that can be spent in one of the following three ways:

1. Co-operatively by Hub and Miner, or
2. By the Hub with a hash lock for pre-image `X`, or
3. By the Miner that found the bitcoin block, but after waiting for six months.

The scriptPubKey for the above conditions in the coinbase is shown in Table 3. These conditions mean that the hub can not spend the coinbase without revealing the pre-image X. This pre-image is included in the construction of payment channels, as we will see in the next section. This use of pre-image in both the coinbase and the payment channel definition guarantees that miners get paid for their accumulated payouts if the Hub defects and spends through the cb_2 branch. We discuss how the miner and the hub don't gain by defecting in Section 5.5.

Fund Transaction

Input	**Output**	
Hub’s UTXO	`2 H  M  2 CHECKMULTISIG`	(f_1)
(Signed by the hub)	`OR`	
	`2 H’ M’ 2 CHECKMULTISIG + Hash(X)`	(f_2)
	(R coins)	

Table 4: Fund transaction for payment channel between hub and miner.

Refund Transaction. Locktime 6 months

Input	**Output**
Fund Tx	`P2WPKH Hub’s address`
(Signed by miner)	(R coins)

Table 5: Refund transaction signed by miner and held by hub.

5.2 One-way Channels

One-Way payment channels between hub and all miners allow miners to receive payouts while consuming a constant size block space. We use one-way instead of bidirectional payment channels as an initial implementation. In future, we can switch to bidirectional channels [10] allowing miners to spend their mining payouts over the lightning network.

Each miner has a one-way payment channel with the hub using a two of two multisig with a time lock of six months. For each payout a miner receives over the payment channel, the hub will charge an agreed upon fees between the miner and the hub. Belcher’s proposal recommends a 0.1% fees for the hub.

5.3 Payment Channel Transactions

The funding transaction includes an input from the hub and an output that can be spent in one of the two conditions shown in Table 4. `H` and `M` are the public keys for hub and miner, they are called the co-operative keys by Belcher. While `H’` and `M’` are alternative public keys for hub and miner, and are called the uncooperative keys in the Belcher proposal.

Just like any other timelocked one-way channel construction, the receiver of the payment, in our case, the miner, creates a refund transaction with a timelock, signs it and sends it to the hub. The refund transaction is shown in Table 5.

If the miner stops responding, the hub can get a refund in six months time. However, the hub can be attacked by sending requests to open new channels and locking up the hub’s liquidity. We resolve this by requiring the hub to open a channel to a miner only after it has contributed enough shares. The threshold number of shares required before opening the channel is a configuration parame-

Payment Transaction

Input	**Output**	
Fund Tx	`2 H  M  2 CHECKMULTISIG`	(p_1)
(Signed by the hub using H')	`OR`	
	`2 H' M' 2 CHECKMULTISIG + Hash(X)`	
	(Hub: $R - earnings$; Miner: $earnings$)	(p_2)

Table 6: Payment channel update transaction sent from hub to the miner.

ter for the hub. The miner will still receive the payouts for the shares generated, it is only that the channel opening is delayed.

On receiving the refund transaction, the hub broadcasts the funding transaction. Once the funding transaction is confirmed, the hub can start sending payouts to the miner. The payment transactions are signed by the hub using the non-cooperating key, `H'`. Table 6 shows the structure of the payment transactions.

5.4 Channel Updates for Payouts

Once a miner mines a share that also meets the current bitcoin difficulty, the miner immediately broadcasts the block to the bitcoin network. The coinbase of this block is shown in Table 3 and can now be spent by one of the following branches:

Co-operative Branch: (cb_1) The miner and the hub by both signing the first branch co-operatively.

Hub Branch: (cb_2) The hub alone, by publishing the pre-image `X`.

Miner Branch: (cb_3) the miner alone, after waiting for six months.

We require that the hub updates the state of all channels, i.e. make payment to all miners. Once it has done so, the miner who found the block signs the first branch of the coinbase transaction and sends the coinbase to the hub. The hub can now redeem the entire payout.

5.5 Hub Defects

If the hub defects and pays itself from the coinbase it uses the cb_2 branch of the coinbase. In doing so, the hub has to reveal the pre-image `X`. With the pre-image available, all miners will use the p_2 branch of their payment channel transactions and close their channels receiving all the payouts earned up to that block.

5.6 Miner Defects

A miner that found the block could chose to not sign the co-operative branch (cb_1) of the coinbase. In such a case, the hub will wait for a timeout period much shorter than the locktime on the miner branch (cb_3) and receive the payout using the (cb_2) branch of the coinbase. In response to this, all other miners will close their channels by signing the (p_2) branch of the payment transaction by using the pre-image `X` included in (cb_2) broadcast by the hub. This will close all channels and require that all these channels are opened again. Such an attack by the miner will hurt miners on the network who have not yet earned enough payouts to amortise the cost of closing the channel.

An attacking miner loses payouts for all the shares it mined since the last block was found by the pool. A malicious miner could contribute a large portion of the pool's hash rate and then refuse to sign the co-operative branch. This will disrupt the functioning of the pool and a well funded censor could be willing to execute such an attack. The only defence here is that the miners re-organise and start a different instance of the pool.

5.7 Hub and Miner Collude

The hub and the miner could collude where they co-operate to spend the coinbase to themselves without requiring that the hub pays all other miners as per the reward schedule. The motivation for the miner is a big payout it can receive. However, such an action by the hub will end the pool and the stream of future profits for the hub.

5.8 DDoS on the Hub

Belcher's proposes the use of multiple hubs as a defence against DDoS attacks. With multiple hubs available on the p2p network, miners will open channels to all hubs and receive payouts from all of the hubs. The coinbase is split between hubs in such a way that if any hub defects, the other hubs can still spend the coinbase and split the block reward between them.

Creating a larger coinbase for multiple hubs scales with Taproot [9, 7, 8]. The solution uses staggered timeouts for hubs to spend the coinbase in case one or more hubs defect or coming under DDoS attacks. The staggered timeouts will result in a large wait for the payouts to be distributed.

We propose a different solution that requires a single hub. The advantage is a simpler channel construction and allow us to build the mining pool without waiting for Taproot activation. The single hub approach required that all miners run a Tor v3 hidden service and publish the public key used to generate the service's address. The public key is included in shares miners publish. The hub can now contact the miners anonymously as a client and therefore avoids DoS attacks on the hub or on the miners hidden services.

5.9 Anonymous Hub Miner Communication

The hub and the miner need to exchange messages at the time of channel creation as well updating the channel state when the hub makes a payout to the miners. The channel management messages are exchanged infrequently as compared to the SHAREs broadcast messages and don't have the same timeliness requirements as the shares broadcasts. We propose adopting a solution that doesn't compromise on the hub's anonymity and makes a trade-off with increased latency of the channel management messages. We use Tor's hidden services to avoid compromising the hub's anonymity.

Miners will run a Tor hidden service next to their mining controller. Once the service is ready, miners include their service's public key in their share headers. The hub can then communicate anonymously with the miners to create payment channels and to update the payment channels. The hub also publishes a well known public key on a publicly accessible location like a github repository or an IRC channel. The hub then participates in the shares broadcast p2p network and identifies a new miner along with the new miner's hidden service address. Once the miner has mined for a certain time period, the hub contacts the miner to set up the payment channels.

To send payouts as channel state updates, the hub sends the updates to all miners using their hidden service address. The hub sends all miners the updates to all payment channels, allowing the miner who discovered a block to verify that the hub has paid everyone, before it sends the channel update transaction to the hub as described in Section 5.4.

6 Future Work

We have not covered the goal of providing tools for building hashrate futures markets in this paper. The initial ideas for the same are captured in a gist[1]. Further, Taproot provides other possibilities for enabling trading of shares published on a p2p network of miners and for increasing privacy of miners and hub.

7 Acknowledgements

Bob McElrath and Chris Belcher for developing some amazing ideas and sharing them freely with anyone who wanted to listen.

References

[1] P2pool. `https://en.bitcoin.it/wiki/P2Pool`.

[1] https://gist.github.com/kulpreet/19927c7188a4224ce2de43efb3c69370

[2] BELCHER. Payment channel payouts: An idea for improving p2pool scalability. `https://bitcointalk.org/index.php?topic=2135429.0`.

[3] CORALLO, M. Compact block relay. `https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki`.

[4] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium* (August 2004).

[5] LEWENBERG, Y., SOMPOLINSKY, Y., AND ZOHAR, A. Inclusive block chain protocols. In *Financial Cryptography and Data Security* (Berlin, Heidelberg, 2015), R. Böhme and T. Okamoto, Eds., Springer Berlin Heidelberg, pp. 528–547.

[6] MCELRATH, B. Decentralized mining pools for bitcoin. `https://www.youtube.com/watch?v=91WKy7RYHD4`.

[7] PIETER WUILLE, JONAS NICK, A. T. Bip 341: Taproot: Segwit version 1 spending rules. `https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki`.

[8] PIETER WUILLE, JONAS NICK, A. T. Bip 342: Validation of taproot scripts. `https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki`.

[9] PIETER WUILLE, JONAS NICK, T. R. Bip 340: Schnorr signatures for secp256k1. `https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki`.

[10] POON, J., AND DRYJA, T. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[11] ROSENFELD, M. Analysis of bitcoin pooled mining reward systems, 2011.

[12] SOMPOLINSKY, Y., LEWENBERG, Y., AND ZOHAR, A. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. `https://eprint.iacr.org/2016/1159`.

[13] VORICK, D. Jute: More scalable, more decentralized proof-of-work consensus. `https://github.com/Taek42/jute`.