Proof of Synchronicity

TRTL001

R. Steady
(Draft 0)

OCT 1, 2018

Synopsis

The social motivation behind mining has become less focused on securing the network as intended and more about accumulating profit. Often the methods of selfish miners have a negative impact on the processing of transactions and user-experience of the network overall. One proposed option is to reallocate the some of the mining reward to reward fully synced daemons.

Assumptions:

- Most miners will usually mine what is most profitable for their hardware.[1]
- Malicious miners can harm the network to enhance profitability.[2]
- More daemons generally achieves lower network latency[3]
- Malicious daemon operators can harm the network to enhance profitability[4]
- Sharing the mining reward between node operators and miners could increase the amount of altruism in the wealth creation process of the network (generating coins) by lowering the profitability of Proof of Work mining on its own.
- Our selfish miner is profit motivated - a maliciously motivated or sufficiently funded attacker will not be stopped by these limits.

Concepts:

- **Miner** - someone who processes transactions on the TRTL Network.
- **Network** - a group of peers running the TRTL daemon software that share the collective transaction history of the chain.
- **Daemon** - the network node of the TRTL Network- allows connecting to peers and syncronizing and verification of the network transaction history.
- **Block** - a segment of transactions processed by the miners on the network.
- **Transaction** - a signed movement of atoms on the TRTL Network.
- **Reward** - Each block processed grants a base reward, plus transaction fees gathered in the transaction creation process.

Problem, expanded: currently 100% of the wealth created in each block goes to the miner. For most people, mining is the primary means of attaining TRTL. Whether a transaction is processed or not, and whether a block is full or not has little consequence to a miner. Mining can give an advantage to those who manipulate timestamps or mine empty blocks.[5] Mining fairness is a key concept in the social element of value assigned to the coins. If coins are easily minted unfairly, the negative effect is a drain on the economy of the network's perceived value; the miner has no incentive to be honest or act in good faith toward the reputation of the network. Miners can also temporarily increase difficulty by renting power and taking advantage of low difficulty, and leaving when the difficulty rises in response. This can cause chain splits as well as irregular block times while the 'natural' hashrate struggles to cope with the sudden increase in difficulty.

---

[1] https://www.cryptunit.com - cryptunit, cryptonote mining profitability tracker
[2] https://github.com/zawy12/difficulty-algorithms/issues/3 - LWMA2 difficulty algorithm, created to negate some of the effects of selfish miners.
[3] https://hackernoon.com/a-thorough-introduction-to-distributed-systems-3b91562c9b3c
[4] https://en.wikipedia.org/wiki/Sybil_attack
[5] https://bitcoinmagazine.com/articles/why-do-some-bitcoin-mining-pools-mine-empty-blocks-1468337739/

Proposal, expanded: By reducing the profitability at face value of mining TRTL, we can reduce the probability of miner based problems like time warps, empty blocks, and pulse mining. Sharing the block reward with daemons has a side effect of encouraging the deployment of more daemons on the network, which lowers the distance between nodes. TRTL has exceedingly fast blocks[6], and often network splits are a result of propagation not being fast enough, and is worsened when hashrate outpaces difficulty suddenly. By increasing the number of nodes, we can assume that this will lower the latency between peers.

Let's assume that the number of miners outnumber the number of nodes 4:1. This is to say that we are assuming that for every 4 miners there is 1 fully synced daemon somewhere. The reality is that with mining pools and web wallets acting as nodes for their users, the number of nodes is likely far less in comparison.[7]

Running a fully synced node currently does not pay any dividend or share of the block reward by default. A user profits nothing from running a fully synced node unless you run a node for others to use, and charge a transaction fee[8]. This lack of compensation has the negative effect of allowing less people to run a daemon, in turn, further increasing the distance between network nodes and increasing network latency potentially.

In a world where miners outnumber full nodes by 4:1 or more, this problem only gets worse over time while the chain gets longer. We can solve this by aligning our reward structure with what we need most as we grow, connectivity. By allowing full node operators to run fully synced nodes and charge a fee on transactions, we saw a slight increase in full nodes, meaning it was possible for people to run full nodes, but the reward to do it wasn't there until we created it.

Note: Due to our fast block times and low transaction volume, fees are often inconsequential in comparison to the base reward, and will be ignored for these models as they are statistically insignificant.

Model Scenarios

Scenario Constants:

- Let's assume there are 5000 participants as part of the network
- Assume a block reward of 29,000.00 TRTL each block
- 1000 participants are running daemons
- 4000 participants are miners without daemons.
- Everyone in this scenario is mining on the same pool, for the sake of the example

[6] https://github.com/turtlecoin/turtlecoin/blob/development/src/config/CryptoNoteConfig.h#L18
[7] http://trtl.world
[8] https://blog.turtlecoin.lol/archives/running-a-public-node-for-fun-profit/

- In this example, the single pool pays every block, and is subsidizing all costs of operation and tx's.

Scenario 1 - All miner reward

This is our current configuration, which favors the miner as the sole winner of block reward.

(29000 / 4000)
(29000 / 0)

4000 miners will net an average of 7.25 TRTL per miner, per block.
1000 node operators get 0 TRTL per block.

S1 Outcome - The reward in this scenario favors the miner, more miners will join the network. Quality of user experience is poor due to high latency, and a high amount of chain splits/reorgs.

Scenario 2 - Percentage Equal

((29000 / 2) / 4000)
((29000 / 2) / 1000)

4000 miners will net an average of ~3.62 TRTL per miner, per block.
1000 node operators get 14.5 TRTL, per node, per block

S2 Outcome - The reward in this scenario favors the node operator, more node operators will join the network. Miners who depend on rapid acquisition and sales will seek other targets. Less chain splits as network latency is lower, blocks propagate faster as a result. With lower internode latency, user experience improves, transaction finality improves.

Scenario 3(A | B) - 2 bit Lucky Validator Nonce

This is an optional game aspect that could be used to increase long term engagement reinforcement through luck, similar to mining. The lottery aspect of the 2 bit lucky validator nonce is a way to give an element of "chance" to winning as a validation-quorum participant. This gives a gamelike incentive for a validating peer to stay online in the chances that they'll be rewarded for holding the full chain and having the correct 2 bit lucky validator nonce. This fattens the pay for those who win, and lessens the efficacy of an attack, giving more chance for it to be noticed in time to intervene. (intervene how? Who knows)[9] If there is a chance the attack will not be profitable even though correctly executed, maybe that's enough to dissuade an attacker. Also, if the value is higher from a sporadic payment rather than a constant trickle of tiny payments (and tiny inputs)[10], perhaps it will keep gambling peers online that wouldn't otherwise be full time peers if they knew they could rely on a steady income stream.

S3(A) Possible Outcome : 2 bit lucky validator nonce is correct, payment address is either included in block template or contract for the next round of payouts.

---

[9] It's unclear what we'd do in this situation. Chain rollbacks are generally frowned upon, and so is centralized means of transaction validation. If this step gets us in trouble with an attacker, we should consider it our own fault and make better rules rather than bending the rules.

[10] Tiny inputs, until RCT/BP's, are an issue to consider, as they fatten blocks.

S3(B) Possible Outcome : 2 bit lucky validator nonce is incorrect. You validate as usual, no pay.

Mechanisms

Requirements:

- The node operator must somehow convey to the network that they wish to be compensated for holding a full node. A public address must be conveyed.[11]
- The node conveying the public address must not, in doing so, compromise the security of the wallet behind it.[12]
- The network needs to form and manage a list of participants
- The network needs to vet those participants as valid full nodes.
- The payouts need to operate on a delay to protect against chain splits draining the reward balance erroneously.
- All nodes validate ITP's by default, whether they seek payment or not
- If this is a smart contract, or handled off chain, the mining work does not need to be touched. If this is all done on the main chain, which is most probable, the set of "winners" will need to be included in the "work" handed to miners so that winners are paid when blocks are unlocked. [13]

Note: This scenario model could use an off-chain solution, such as DHT[14] or sidechain as means of distributed consensus and tracking of non permanent data, like the list of addresses seeking payment for the node operator role of each top block.

Concepts:

- **validators** - fully synced daemons who manage a list of the peers on the network who have announced an ITP. all unconfirmed tall peers are validators.
- **validation-quorum** - a group of peers who: each block, generates a "validation checksum" consisting of the first and last block hash characters from 4 random block heights (challenge heights) selected during the previous block, which is then combined with the peer nonce to make the 10 bit Validator Challenge String
- **2 bit lucky nonce** - a 2 bit lucky nonce that can be created by the user or random.[15]

---

[11] Could be as simple as *./TurtleCoind --announce*
[12] A daemon's IP is public. We don't want to attach a cleartext wallet address to this daemon that can be pattern matched and linked to the user. We should generate an address and keys for the user instead. If the user doesn't voice a need for payment, node is not payable.
[13] A miner likely cannot be forced to pay winners, validators must control this if possible by rejecting blocks that aren't paying winning validators. Needs some type of *settlement layer*.
[14] https://en.wikipedia.org/wiki/Distributed_hash_table
[15] Probably wouldn't be a problem with a peer using their own lucky number here.

- **10bit Validator Challenge String** - is a 10bit challenge response verified by validation-quorum to verify a node is real[16] and is not a spoofed daemon or fuzzer. This should damage probability of a rogue validator gaming the validation system as it adds a bit of randomness to payouts as only those with the correct 2 bit peer nonce get paid. (NEEDS WORK)[17]
- **Announce** - A daemon will submit, for each block period it participates in validating, an Intent-To-Participate to becomes part of the validation process.
- **ITP** - Intent to participate consists of: Public Address, 2 bit validator nonce, 4 challenge heights.[18]
- **Tall peers** - peers that have established themselves as having correct challenge heights.

Timeline

On the network - Block 900,000 is created, hash is recorded by validators
On the network - Validation-quorum pays rewards for Height 899,979 (always pay in arrears to protect from chain splits)
On the network - Validation-quorum converts block hash 900,000 to decimal, and constructs 4 heights before block 899,979 and records them as challenge heights for block 900,000.[19]

At home - Daemon is launched with an announce address, begins syncing
At home - Daemon generates a bespoke public address, relays keys to user [20]
At home - Daemon finishes syncing
At home - Synced daemon announces ITP request to validation-quorum for block 900,000
On the network - Validation quorum sends back an ITP response ACK if challenge heights are correct, ignores if wrong/late.[21]
On the network - validators record 2 bit lucky nonces and their public addresses
On the network - somehow validator quorum selects two random bits for the winner(s) without use of an oracle[22]
On the network - Validator quorum iterates through list of participants, removes duplicate duplicate ITP's and peers with incorrect challenge heights [23]
On the network - Add tall peers as validators in validation quorum for block 900,001 (tall peers have passed challenge height check)

---

[16] If a fuzzer hooked up to the network can correctly guess the first and last characters in the block hash to 4 random check heights, just fucking give it to them. It will never happen. Probably is a synced node.
[17] This needs work because even though a lottery system only penalizes those paying for the resources to attack this, it could be done better.
[18] Looks like *[98 char TRTL add.] + [2 bit lucky nonce] + [checkheight{1..4}]* with no spaces between sets.
[19] We need a way to derive randomness, use blockhash converted to decimal to derive check heights from before payout period we're validating.
[20] This forces the user not to reuse an existing address, which could compromise anonymity.
[21] Purpose of the ACK is for the daemon to return some type of response to the user so they receive confirmation of being in the system.
[22] This could be expanded on.
[23] Something has to be done to discourage a peer from inserting themselves in the record numerous times covering every lucky nonce possibility. Last Address Seen seems to be the best way to combat this while keeping pool size down.

On the network - Block 900,001 is created, hash is recorded by validators
On the network - Validation quorum pays rewards for Height 899,980, or, as block 900,001 is mined, it pays winning validators who selected the correct lucky nonce. (lottery system)[24]

...

Needs, Considerations, TODO

- This proposal needs some way to take into account peer density, meaning if you can reach many peers and they're all 0ms away, you're probably not a benign participant, or useful.
- Peers that are too numerous while being too close could potentially worsen round trip time for data reaching nodes on the "last mile"

[24] The process continues..