Layer 2 blocks

April 18, 2022

Abstract

This is a rant. Having translated Zero to Monero v2 to portuguese (WIP), i am hoping, that someone will explain to me why layer 2 blocks are *not* a solution for the spacial scalability of Monero. In a way that i will understand, making the topic for me mute. And since i regard this question, others could too, therefore it is reasonable to have a section in getmonero.org that explains its pros and cons. The translation is here : `https://raw.githubusercontent.com/UkoeHB/Monero-RCT-report/master/Zero-to-Monero-2-0-0-pt.pdf`. I also have a version hashed in the btc blockchain, which i can sign. I guess it makes no sense to be a speaker and rant about this for half an hour, i guess it would be embarassing for the audience. The best case for me, is for it to be implemented in a future version of Monero and letting me write some loc's too. The worst case is relative, but do please, at least answer me (`dollner@gmx.de`). I don't know yet if i will make it to konferenko, and great that it is on a weekend, thanks.

Crypto-currencies have no value. It is important to recognize and understand this aspect. No one can eat a digital coin nor do anything with it in the real world. The only value they have lies in their utility. Wich remains always in the digital realm.
Assuming that the currency is well established and has undergone a certain amount of network effects, amount of users, amount of full nodes, a safely high global hashrate, etc. Then the price of a digital coin tends to the cost of mining it. For advocates of sound money this is somewhat twisted; "basically the price of a coin equals its cost of creation, and its value lies in it's digital utility."
Therefore having transactions that are not even bound to this logic are even more departed from the *notion of value*. This is, in Bitcoin the lightning network, might even be ubiquitous in the future. However only the opening and closing of the channel is mined, thus only thouse transactions are bound to a cost, wich in turn gives the crypto-currency itself a price.
Let the contrary be true, all current blocks are almost empty of transactions. Because the global need for transacting is currently being done over the lightning network. The only transactions being recorded in the last block, are the multi-sig ones that close channels. Everyone is happy transacting and all is good, the crypto-currency has value because its being used. But what is the overall cost of transacting, from the perspective of the miners? And therefore what is the price of the crypto-currency?
Following this line of thought, the importance of zero confirmations is paramount. Assuming that every valid transaction that is relayed to all full nodes is always mined, then why would a recipient care if it is already in a block or not? So if a vendor sees the zero-conf payment ocurring he is happy. The only problem here are spam-attacks or moments in which the network has a spike in the demand for transactions. Because the mempool of the individual nodes could get overflowed, therefore the transactions don't stay in the mempool, being effectively lost. Thus the transaction has to be send out to the network again by the sender.
Complexity lies in the eye of the observer. In problems that are np-hard/complete the complexity in terms of exponential growth in time or space required to solve a problem of input size N, make the problem *intractable*. That means, for a rising N there is no known polynomial time (or space) algorithm that solves the problem. When building heuristics for such problems, what often happens is that the complexity is passed along from one place to another like a hot potato, but it does not go away. The goal of this *rant*, for me is to understand why slave blocks are not a solution to the scalability of a blockchain, given visa-like transaction volumes. Here scalability refers only to the requirements of *unprunable* storage space. So the form of a layer one block, master block, goes as follows :

The normal stuff of a Monero block, like previous block hash, miner transaction, etc and then :
hash of first slave-block, followed by all key images from that slave block. Hash of the second slave-block, followed by all key images from that slave block. Hash of the third slave-block, followed by all key images from that slave block, etc until the n-th slave-block.

Where a slave-block has the number of the master-block it belongs to and a bunch of transactions.
A slave-block is from now on denoted as *s-b*, and a master-block as a *m-b*. A *s-b* does not have any subsidy. So when a coin starts, the genesis *m-b* has no transactions besides the miner transaction. As the first *m-b*'s are mined the amount of transactions increases. Let each *s-b* have the same maximum amount of transactions. And let that maximum be the amount of mixins from the ring signatures. So while the entire blockchain fits on every joe's hard drive without any problem all is good. The amount of transactions per *m-b* is not so big, and all joe's are like : "the *entire* chain fits on my hard drive, and it will fit for the next two years, if the same transaction volume continues." While this is true then all *s-b*'s are kept with their *m-b*'s . When this starts to be like : "the *entire* chain *does not* fit on my hard drive, and every t amount of time i need to buy a new hard drive, if the same transaction volume continues." Then it is *assumed* here that the relevant parameters for tackling the spacial scalability in

this cripto-currency have been met. These are : amount of full nodes, mempool size of each full node, transaction volume, bandwith, computing power of each full node, and most importantly the hash rate. The hash rate is considered paramount because the security of the blockchain would be in *suspension*. Or also implicit. What is suggested is that from a given point forward, all full nodes only keep the *m-b*'s, from the genesis *m-b* up to a certain *horizon start*, and after that a h amount of *m-b*'s together with their *s-b*'s (*horizon*). The question at hand is :

$$\lim_{t \to +\infty} (c * \epsilon) + horizon \qquad [1]$$

where ϵ is the size of the *m-b*'s without the prevention of the double-spend information, here only the hashes of the *s-b*'s. And $c * \epsilon$ is the size including the key-images for every *s-b*. And *horizon* is a size considered a constant wich is negligible as time goes on. As time goes on, and the volume of transactions reaches the desired goal of 50k tx/s, how small can $c * \epsilon$ be for a given blockchain? In other words for the total amount of transactions in a blockchain how big must the chain composed only by the *m-b*'s be ? Let the chain already have a horizon with a default length of *m-b*'s for every node. Then sending out a transaction alone is only valid if its inputs are within this horizon. Because all of it's necessary data is by default known by the miners of the network.
If not then also the *s-b*'s, one for every input have to be present. The miner takes the transaction, and for each input and it's *s-b*, checks if at the *s-b*'s respective *m-b*, exists the hash of this *s-b*. If the hash of the *s-b* exists at it's *m-b*, then that input must be valid. And if the key-images of this transaction have not yet apeared in any *m-b*'s so far, then there is no double spending going on. Furthermore, everything related to this transaction is implicitly valid, because it has been validated before. The pre-image resistance of the hash, and it's hash function makes it *impossible* to know, what data was hashed, and therefore the hash of the *s-b* is information teoretically secure/private. There is even an infinite amount, of distinct inputs that produce the same hash, and they are collision resistant.

And this is where the complexity apears :
lets say that every tx only has *one input* and two outputs. Every tx has a rough size of 2Kb, the amount of tx's inside a *s-b* is 11. Let a hash of a *s-b* be 32 bytes, and also 32 bytes for every key-image. The size of a *s-b* is 11*2Kb = 22Kb. Then the space requirement at 50k tx/s for a *m-b* outside the horizon is :
120s(50 000tx * 32bytes) + (545454h * 32bytes)= 192000000 bytes + 17454528 bytes = 192000 Kb + 17454 Kb = 192 Mb + 17 Mb = 209 Mb . Where 545454h is the amount of *s-b* hashes, and comes from : (120s*50 000tx)/11
50000tx/s → 192000000 bytes + 17454528 bytes = 192 Mb + 17 Mb = 209 Mb
5000tx/s → 19200000 bytes + 1745440 bytes = 19 Mb + 1 Mb = 20 Mb
500tx/s → 1920000 bytes + 174528 bytes = 1,9 Mb + 0,1 Mb = 2,0 Mb
50tx/s → 192000 bytes + 17440 bytes = 0,19 Mb + 0,01 Mb = 0,2 Mb

lets say that every tx only has *k inputs* and two outputs. The size of a tx will rise logaritmically, but let's keep the size of 2Kb to make the following calculations easier. Then the space requirement at 50k tx/s for a *m-b* outside the horizon is :
120s(50 000tx * 32bytes * k) + (545454h * 32bytes)= k * 192000000 bytes + 17454528 bytes.
50000tx/s → k * 192000000 bytes + 17454528 bytes = k * 192 Mb + 17 Mb
5000tx/s → k * 19200000 bytes + 1745440 bytes = k * 19 Mb + 1 Mb
500tx/s → k * 1920000 bytes + 174528 bytes = k * 1,9 Mb + 0,1 Mb
50tx/s → k * 192000 bytes + 17440 bytes = k * 0,19 Mb + 0,01 Mb

The space requirement at 50k tx/s for a *m-b* inside the horizon is :
209 Mb + 120s(50 000tx * 2000bytes) = 209 Mb + 12000000000 bytes = 209 Mb + 12000000 Kb =

[1] the limit has t, without t being used. Thats great this is a rant!

209 Mb + 12000 Mb = 12,209 Gb . And since each one input transaction inside the horizon still has to prove itself with the respective *s-b* it's : 12,209 Gb + 120s(50 000tx * 22Kb) = 12,209 Gb + 132000000 Kb = 12,209 Gb + 132000 Mb = 12,209 Gb + 132 Gb = 144,209 Gb

Which sounds hopeless and awfull at the time of this writing. Reducing the tx amount :
50000tx/s → 144 Gb
5000tx/s → 14,4 Gb
500tx/s → 1,4 Gb
50tx/s → 140 Mb

This is for tx's that have only one input, if there are k inputs its :
12,209 Gb + 120s(50 000tx * 22Kb * k) = 12,209 Gb + (k * 132000000) Kb
50000tx/s → 12,209 Gb + (k * 132) Gb
5000tx/s → 1,220 Gb + (k * 13,2) Gb
500tx/s → 122 Mb + (k * 1,32) Gb
50tx/s → 12,2 Mb + (k * 132) Mb
(the term 209Mb also changes, but it's ignored here, and left the same.)

As an adicional argument to why do this, its also that just because storage space is available and very cheap it does not need to be used. If in the future Gb's are regarded as Mb's, and every joe could keep all tx's in a normal blockchain without any pruning whatsoever it would still be arguably more *aesthetic*. However another argument can be found in terms of privacy/security. Say the blockchain has n blocks, and a new node starts to make the full sync. It is assumed that if that monerian has any private keys to his funds then he also has the respective *s-b*'s. So in fact the other *s-b*'s don't matter to him, as he only cares about his funds. This means that chain analytic companies would have to keep all the blocks all the time, wich may or probably is not a problem, since they would be always syncing and have enough storage space. However if a new startup wants to analyze the chain then it needs to get all the data from somewhere. Since if there is no data then there is also no data to be analyzed. Wich is sort the extreme of keeping it simple, regarding that cyber security *likes* simplicity.

Assuming that the numbers that where ran before in this rant, are attractive... So the question that arises in such a case is how big does the horizon need to be such that the blockchain is secure ? Because a 51% attack does not start a few blocks after the genesis block, it starts at the current latest block. What other risks happen? Why is this not an option ? Is it not that the longest chain with the biggest comulative difficulty keeps the right to be the true chain? Therefore the initial block download must be fine, because what difference does it make for the nodes to sync blocks that until the current i-th block make sense ? Why does one have to recalculate the dificulties and verify that all the miners subsidys plus the fees make sense, until the current i-th block? Let a chain in total be 100 blocks long, with a horizon of 33 blocks. Is it the only problem, if i *start syncing*, that because i can not verify trustlessly what the true horizon is, that syncing is impossible ? Well then, i guess syncing the horizon first is not possible, because there is no frame of reference...
But if done backwards would be possible? first syncing a horizon of blocks and then starting to sync at light speed just the hashes of the *m-b*'s, from the start of the horizon all the way down to genesis!? Is the IBD a teoretical impossibility ?

In the strictest sense it is not, because if 2 nodes are *always* online, then they could keep up with the story...!

Are there teoretical impossibilities ?