Mempool optimized fees, and the correlation between user costs, miner incentives, and block capacity

Karl-Johan Alm <karl@dglab.com>
DG Lab

October 31, 2017

Abstract

We have investigated the correlation between fees, miner incentives, and block capacity in Bitcoin. In particular, we look at how fee estimators can be optimized to more precisely pinpoint the fee rate required to get into blocks in the near future, as well as how better overview of mempool state can be helpful in a variety of (sometimes hostile) situations.

Throughout the report, we are deriving results via mempool statistics, sometimes from models and sometimes from simulations on real data. In Bitcoin Core, and many other wallets, fee estimations are done exclusively using past block statistics, and the mempool state is ignored completely.

While the results will in some cases undershoot, our results show that proper use of RBF gives great potential to taking the mempool into account during fee calculation.

A consequence of optimizing fees is a potential drop in revenue for miners. Currently, user fees comprise about 11% of miner revenue, and with optimizations this number would drop even lower. With subsidy halvings every four years, it is important to address how miner profits and user costs correlate, in particular how they are related to transaction throughput.

We investigate consequences of block capacity changes, and conclude that a capacity increase is not an adequate solution at this point in time, as it will only minimally affect the miner revenue at the cost of less security, more centralization, and the incentivization of a remine attack in low peak situations. We note that a higher block capacity does, however, make certain hostile attacks more difficult and costly.

We also investigate the consequences of an empty mempool once subsidy has become a less significant part of the miner incentive. We conclude that the network benefits greatly from having a mempool with a backlog, and that this is an inherent security feature of Bitcoin that should be preserved.

CONTENTS

CHAPTER

ONE

INTRODUCTION

This report investigates the correlation between user cost, miner incentive, and block capacity. The viability of using the mempool to optimize fee rates for transactions, as well as using the mempool to respond to certain attacks is explored.

It begins in Chapter 2 by building a model for the mempool and block generation. It explores how the block capacity would change given certain assumptions, and how this would relate to various constraints (2.1), such as a minimum required bitcoin per block fee for the miners (2.1.1), or a minimum profit expressed in USD (2.1.2), as well as how this would change with time as the subsidy drops. The chapter ends with an examination of the consequences of a blockchain system where all transactions are mined at all times, and some important security related problems are derived (2.2).

In Chapter 3, the opposite approach is examined, where the mempool is considered a market. Here, the viability of certain attacks on layer 2 tech is discussed (3.1.3).

In Chapter 4, a fee rate optimization using the mempool is discussed, and conclusions are drawn in Chapter 5.

CHAPTER

TWO

BLOCK AND MEMPOOL MODEL

This chapter builds a model around the mempool, block capacity, and user (transactor) cost. Note that much of it is assuming that the mempool is empty, and that every new block consumes the contents of the mempool at the given time.

2.1 Block size, miner profit, and user cost

Let $\wp$ be the collection of transactions $\vec{\tau}$ in the mempool at some given point in time. The two functions $\omega(\tau_i)$ and $\mathrm{FR}(\tau_i)$ give the weight[1] and fee rate of the transaction τ_i.

$$y = \sum_{\tau \in \wp} \omega(\tau) \tag{2.1}$$

is the aggregate weight of all transactions in the mempool, and

$$y_{\mathrm{FR} \leq x} = \sum_{\tau \in \wp} \omega(\tau) \forall (\mathrm{FR}(\tau) \leq x) \tag{2.2}$$

is the aggregate weight of all transactions τ which satisfy $\mathrm{FR}(\tau) \leq x$ for some x, i.e. all transactions with a fee lower than or equal to x.

Using a naïve, perfect knowledge no-RBF universal fee estimator[2], the fee rate becomes

[1] SegWit unit for size, sometimes written as "WU" for "weight unit(s)"; it is often approximated so that 1 byte = 4 WU in this report, even though this is not entirely accurate. ω is also used instead of "WU" to express weight units in this report, but this is by no means conventional.

[2] Naïve because it assumes no competing transactions will appear; perfect knowledge because it sees the entire mempool of the network; no-RBF because it would otherwise bump its fee repeatedly rendering the algorithm essentially worthless; universal because all other transactors on the network are assumed to use exactly the same algorithm.

$$f(y, Z) = \frac{1}{4} 1.1^{\lfloor \frac{y}{0.95Z} \rfloor} \text{ satoshi}/\omega \tag{2.3}$$

where Z is the block capacity in weight units.[3]

The per-block profit without subsidy simply becomes

$$f(y, Z)Z \text{ satoshi} \tag{2.4}$$

which, for a just-full mempool[4] simplifies to $\frac{1}{4}Z$ satoshi.

In other words, a 4 MWU = 1 MB block will on average get 1 million satoshi $\approx$ \$55 USD[5].

In contrast, the current miner profit from subsidy alone (i.e. excluding all transaction fees) is roughly \$65,000 USD.

We will return to $f(y, Z)$ but before we do, we will take a look at two cases related to miner profit: the first case is for a fixed BTC income per block, and the second case is for a fixed USD income per block.

2.1.1 10 BTC per block assumption

We explore how bitcoin will change when the mempool is optimal w.r.t. the blocks, i.e. no transaction backlog, and every block filled to capacity.

If we make the assumption that miners desire 10 bitcoins per block after the next halving in $Y = 2020$ (they get 12.5 as of this writing), we begin by defining

$$r(s) = 10.0 - s \tag{2.5}$$

as the fee required per block after subsidy s, where

$$s = \frac{12.5}{2^{\lfloor \frac{Y-2016}{4} \rfloor}} \tag{2.6}$$

(Y being the current year) giving us $r_s(s) = 10^8 r(s)$ as the fee required in satoshi.

The satoshi/weight (assuming a transaction takes up 1200 weight) becomes

$$F(p, c) = \frac{p}{1200} \cdot \frac{10^8}{c} = \frac{10^8 p}{1200c} \tag{2.7}$$

where p is the transaction fee in US dollars, and c is the current USD/BTC price. E.g. for $p = \$.20$, $c = \$5500$ (20 cents at current price of \$5500 USD/BTC), $F(p, c) = \frac{10^8 \cdot 0.2}{1200 \cdot 5500} = 3.03$ satoshi/ω, or 0.76 satoshi/b.

The weight required at the given fee $F(p, c)$ to achieve the goal of $r_s(s)$ satoshi for the block is expressed as

$$Z(s, p, c) = \frac{r_s(s)}{F(p, c)} = \frac{1200(10.0 - s)c}{p} \tag{2.8}$$

[3] The way this function is derived is given in Appendix A on page 27.

[4] Transactions are always mined at the next block, and blocks are always exactly full.

[5] Taking 1 BTC = \$5500 USD

With $s = 6.25$ (2020 through 2023), $p = \$.20$, $c = \$5500$, we get that the necessary block size in bytes is

$$\frac{1}{4}Z(6.25, 0.20, 5500) = \frac{1}{4}\left(\frac{1200(10.0 - 6.25)5500}{0.20}\right) = 30937500,$$

i.e. roughly 30 MB.

This hits a maximum when $s = 0$ (no subsidy) at roughly 80 MB.

If the users aren't willing to pay the 20 cents, the block size increases proportionately, where if p is halved, the block size is doubled (and conversely, if p is doubled, the block size is halved). If the price of bitcoin relative to USD increases, the block size increases in the same fashion, and if it drops, the block size drops. E.g. at \$1,000,000 USD/BTC, the block size would cap at $\frac{1}{4} \cdot \frac{1200 \cdot 10 \cdot 1000000}{0.20} = 14$ GB (but the miners would get \$10 million USD per block).

2.1.2 $65,000 USD per block assumption

We approach the problem from the point of view of a given revenue in USD[6]. We begin by rewriting $r(s)$ defined in Equation (2.5) as

$$r(s, c) = \frac{65000}{c} - s \tag{2.9}$$

e.g. for $c = \$5500$ USD/BTC, $s = 6.25$ BTC, we get $r(s, c) \approx 5.57$ BTC. The miner revenue would simply be $\frac{65000}{c} \approx 11.82$ BTC.

Inserting this into Equation (2.8) we get

$$W(s, p, c) = \frac{r_s(s, c)}{F(p, c)} = \frac{10^8\left(\frac{65000}{c} - s\right)}{\left(\frac{10^8 p}{1200c}\right)} = \frac{1200}{p}(65000 - sc) \tag{2.10}$$

The block size $Z(s, p, c) = \frac{1}{4}W(s, p, c)$ grows inversely proportionately to p (the price users are willing to pay in fees for a transaction), and shrinks with subsidy s and price of bitcoin c; however, as the subsidy drops, the impact of the price of bitcoin drops proportionately until it stops having any effect[7]. Table 2.1 shows a matrix for the next subsidy halving ($s = 6.25$) and (right-most column) for the case $s = 0$.

We can also display the case $s = 0$ with a variable profit value, so that $r(0, c, \omega) = \frac{\omega}{c}$, as shown in Table 2.2.

[6] Here we picked the current subsidy of 12.5 BTC in USD at \$5500 USD/BTC.

[7] This is not entirely obvious, but since we are defining user fee in terms of USD, and miner profits in USD, the amount of satoshi paid for a transaction will change, but proportionately to the miner profit. The equality $65000 \cdot \frac{1}{c} - s = \frac{p}{1200} \cdot \frac{1}{c} \cdot Z$ has $\frac{1}{c}$ (BTC per USD) on both sides, which cancel out when $s = 0$, leaving $65000 = \frac{p}{1200} \cdot Z$, and $Z = \frac{65000 \cdot 1200}{p}$, which is unaffected by changes in c.

	Block sizes for $s = 6.25$					for $s = 0$
	Price USD/BTC					Any price
Avg fee (USD)	\$6k	\$7k	\$8k	\$9k	\$10k	\$$x$
\$0.10	77 MB	61 MB	43 MB	25 MB	7.2 MB	186 MB
\$0.20	39 MB	30 MB	21 MB	13 MB	3.5 MB	93 MB
\$0.50	16 MB	12 MB	8.9 MB	5 MB	1.4 MB	37 MB
\$1.00	7.9 MB	6.1 MB	4.3 MB	2.5 MB	0.7 MB	19 MB

Table 2.1: The block size for various average transaction fees vs the price of bitcoin, with a fixed subsidy $s = 6.25$ (2020-2023) and (right-most column) when $s = 0$, at which point the price no longer matters.

	Block sizes for $s = 0$				
	Miner profit threshold				
Avg fee (USD)	\$20k	\$50k	\$100k	\$250k	\$500k
\$0.10	57 MB	143 MB	286 MB	715 MB	1.4 GB
\$0.20	29 MB	72 MB	143 MB	358 MB	715 MB
\$0.50	11 MB	29 MB	57 MB	143 MB	286 MB
\$1.00	5.7 MB	14 MB	29 MB	72 MB	143 MB

Table 2.2: The block size for various average transaction fees and various miner profit thresholds, given a zero subsidy (s).

2.2 Interruptions and consequences of mempool exhaustion

With no subsidy left, a miner will not spend their resources on finding a block unless the fees in the publicly known unmined transactions are at minimum higher than the electricity costs of doing so.

As such, the higher the block size, the more likely it is that the chain will have *interruptions*, where no block is being mined by anyone. Miners would switch their equipment to some competing chain with the same proof-of-work (if any such chain existed) or put the mining equipment in "economy mode" to minimize costs.

This minimum requirement is additionally different per region, because electricity cost is different. As such, this state will lead to centralization of mining power, to where a region with lower electricity costs will represent a disproportionately high portion of the hashpower.

2.2.1 The for-profit and essentially free reorganization attack

This presents a major security problem. It would be profitable for a miner to purposefully drop the last block (if it was mined by someone else) and remine it, to gain the transaction fees from it, while the rest of the network is waiting for transactions to satisfy the above requirement. The miner could then mine an extra block on top of his remined one and the two blocks would replace the current tip as the chain with the highest amount of work.

He could do this for an arbitrary number of blocks if he has enough hashpower, because everyone else will be waiting for transactions to cover their expenses after every "honest" block.

The difficulty of the chain would drop as well, because miners were not mining blocks every ten minutes, further reducing its security, and further making the above attack more easy to pull off.[8]

To counter this attack, miners of a block would need to keep mining an empty block so that their chain tip remained protected. This would only work if no other malicious miner had more hashpower, and even if, the miner would be mining at a loss. To prevent this, miners would be enticed to form coalitions or enter into contracts with other miners to promise to protect each others' blocks, leading to a higher degree of centralization.

2.2.2 Mining cost vs transaction (fee) throughput

One big argument against the above scenario is that the cost to mine a block grows with time, and this growth only needs to be slower than or equal to the

[8] Assume that most miners need to wait 60 seconds before they are able to mine at a profit. The average time to find a block would then be 9 minutes, which would be a reduction of 10% in hashpower, and consequently a 10% reduction in network difficulty compared to the available hashpower.

transaction (fee) throughput for the miner to be profitable. I.e. if a miner finds a block after a few seconds, the revenue would only need to cover the few seconds worth of electricity usage. Because electricity cost varies, this would mean some miners would profit from mining earlier than others. That aside, this depends on the arguably unstable condition that transaction (fee) throughput is *constant at all times*, which it is not; it is experiencing highs and lows and this trend will most likely grow in scale with adoption. In other words, miners will still benefit from *not* mining during low peak periods, for the case where the blocks have sufficient capacity to include all transactions at high peaks.

One might assume this is an argument for dynamic block sizes, but this would accomplish nothing. A static block size that would fit all transactions at peak time would be just as effective even if only a fraction of its capacity were used during the low peaks.

Assuming Bitcoin is successful, one might argue that none of this would ever pose a problem, but if the blocks are large enough to include *all transactions at all times*, there will inevitably be periods of high-peak activity and periods of low-peak activity (VISA today averages roughly 2,000 transactions per second, but has a capacity of 24,000[9]). It would only take one low-peak period to encourage the voluntary reorganization attack mentioned above by a miner.

2.3 A note on subsidy halving

A perhaps unusual amount of emphasis is placed on the subsidy halving and its effect on the dynamics of Bitcoin scaling in this report. Since the subsidy will not vanish entirely until roughly the year 2143, this may seem like making a mountain out of a pebble. However, it should be noted that due to the nature of the subsidy halving process, the highest impact will be seen earlier — for instance, the subsidy will drop by a total of 596 satoshi in the entire final 43 year period 2100 - 2143; it will be lower than 0.001 bitcoin from 2072 and onward, and it will drop below 1 bitcoin per block as soon as 2032, 15 years from now.

See Appendix B for additional details on how subsidy halving and fees might correlate.

[9] https://usa.visa.com/run-your-business/small-business-tools/retail.html

CHAPTER

THREE

TRANSACTIONS AND THE MEMPOOL AS A MARKET

This chapter takes the approach of looking at the mempool as a market, and describes issues with this approach, such as overly conservative fee estimators (3.1.1), and issues related to the user interface and user experience in software (3.1.2). It also investigates the viability of a specific form of attack on time locked smart contracts such as the Lightning Network and atomic swaps (3.1.3).

3.1 Transaction throughput and mempool market

We return to $f(y, Z)$ originally defined in Equation (2.3) on page 5.

$$f(y, Z) = \frac{1}{4} 1.1^{\lfloor \frac{y}{0.95Z} \rfloor} \text{ satoshi}/\omega \tag{3.1}$$

We define a transaction throughput Δy given in weight per block[10]. Assuming no mempool backlog,

- If $\Delta y < Z$, the next block will not be full. It will be at $\frac{\Delta y}{Z}$ capacity where $y = \Delta y$, and the miner will get $\frac{1}{4}\Delta y$ satoshi according to Equation (2.4).
- If $\Delta y = Z$, the next block will be full and the miner simply receives $\frac{1}{4}Z$ satoshi.
- If $\Delta y > Z$, the next block will be full. There will be a growing mempool backlog for as long as this remains the case, where

$$y_{t+1} = y_t + \Delta y_t - Z \tag{3.2}$$

[10]Strictly speaking, it is weight per 10 minute interval, disregarding discrepancies in the time between blocks, which varies quite a bit.

In the last case above (Equation (3.2)), we have appended a time subscript t to the mempool size y and the corresponding transaction throughput Δy. In the simplest case, Δy remains fixed for all t, and y_{t+1} simply becomes $(\Delta y - Z)t$, after t blocks.

As $t \to \infty$, so does $y \to \infty$, and consequently $f(y, Z) = 1.1^{\lfloor \frac{y}{0.95Z} \rfloor} \to \infty$. In other words, we approach infinitely high fees if the transaction throughput per block exceeds the block capacity.

We note here that, if RBF was enabled, every participant would repeatedly bump the fee of their transaction as soon as $y > 0.95Z$ in order to beat the bottom 5% threshold. Even at $t = 0$, this bumping would continue without bounds moving toward ∞ in the same fashion, only faster. The only real reason why t matters at all is because each block has a number of ever growing "losing" transactions in the mempool that were created too early to catch the real threshold.

As the fee rate increases, our willingness to transact decreases. It becomes clear that $f(y_t, Z)$ breaks down as $\Delta y_t \geq Z$.

We thus redefine this as $f(x, y, Z) = \min(x, f(y, Z))$, where x is the maximum fee rate the user is willing to pay.

We can imagine grouping x into ranges like "emergency", "high priority", "normal priority", and "low priority", where the ranges vary depending on what people are paying in fees right now. Rational miners would start from the top and include transactions until they hit the block capacity.

This approach works in standard cases, but has some problems.

3.1.1 Overly conservative fee estimators

One main problem is when fee estimators are suboptimal and/or overly conservative. With RBF support, the ability to fine-tune fee rate is better than ever, but this is not taken advantage of by most, if any, wallets at this time.[11]

Related to this is that the fee estimators currently do not take into account the amount being sent. Paying a $1 fee if you are sending a thousand dollars is more acceptable than if you are sending $2, and aiming for a close-cutting fee in the latter case would most likely have proportionately acceptable consequences.

3.1.2 User interface

Another problem is that a lot of users don't realize they actually set the fee rate themselves. This ties in with the previous problem, in where wallets tend to be overly conservative in their estimates[12]. Better education of users and better

[11] Bitcoin Core 0.15 will use economic fee estimation for transactions which have RBF set, but that's the only case of a wallet we've seen so far. Peter Todd's OpenTimestamps (https://opentimestamps.org/) is one of the few systems using it today.

[12] Possibly to prevent a storm of angry users asking why their transactions are not being confirmed.

UI in wallet software would possibly remedy this problem, especially if wallets turned RBF on by default[13].

We will expand on this in Chapter 4.

3.1.3 Time-locked smart contract expirations

Many smart contracts, such as the Lightning Network, depend on the ability to detect and react to a malicious transaction, usually by transferring the channel balance to oneself. A too intensive mempool market might allow an attacker to steal funds from a channel. We will describe the attack, and examine its viability below.

Old channel state attack

- A channel with an ℓ block locktime[14] is established between participants A and B with c bitcoin funded by A, through a funding transaction τ_0 with an initial state/transaction τ_0 of $c \to A; 0 \to B$.
- A pays d bitcoin for some service by transferring funds to B over the channel, creating state/transaction τ_1 of $c - d \to A; d \to B \quad (d \leq c)$.
- A makes a malicious attempt to close the channel by broadcasting τ_0, then proceeds to spam the network with lots of transactions.
- B attempts to claim the funds, but the mempool is overcrowded. A continues to spam the mempool, adjusting their fees to always keep B's transaction from being mined.
- After ℓ blocks, if A has succeeded in continuously making enough transactions to bump B's claim transaction out of any blocks, they can now attempt to claim the funds before B manages to do so.
- If they succeed, they have made $d - f^A$ in profit, where f^A is the sum of the fees paid to maintain the mempool fee rate required to keep B's claim transaction out.

This depends on a number of things: d must be so large that A would benefit from doing this, despite spamming the network for the duration of the locktime. It would be more beneficial to A if the mempool was already under heavy load from external sources, assuming τ_0 is confirmed reasonably fast.

B can counter using RBF by simply tracking the mempool and ensuring their transaction makes it into the block. B is also at a great advantage due to

[13] This would allow users to fix "too low fee" mistakes, but they could still end up paying a very high fee by mistake, in which case RBF would not help.

[14] A feature which states that, once the transaction has been confirmed in a block, it cannot be spent until at least this number of blocks have been mined on top of the block it is contained in. This lets users put a "hold" on transactions for a given period, so they have time to react to it when it appears in the blockchain.

the fact B simply needs one transaction to go through, while A needs to ensure the whole mempool is "covered" without letting B's transaction in.

The higher the locktime count ℓ is, the harder and more costly it is for A to perform this attack.

The risk offset is usually a deterring factor to attempting this attack, as A would gain d bitcoin whereas B would gain c, where $c \geq d$. If $c - d \approx 0$, the only financial risk for A is f^A, especially if A's identity is unknown.

The attack works best for channels with larger amounts, but it works with any number of B's, i.e. A may close *all* their channels in this manner and try to keep all B claim transactions out (selectively abandoning the ones which would require a too-high fee).

Viability

We will first simplify by assuming that B makes one transaction and does not bump it to respond to A's spam transactions.

Let τ^B be the claiming transaction $x \to B$, which A must not allow to confirm for their attack to succeed, and $\vec{\tau}_t^A$ be the collection of spam transactions emitted by A at time t.

y_t is the transaction weight at time (block) t, for transactions $\vec{\tau}_t$, assumed to be at their highest tolerated fee rate, and Δy_t is the influx of transactions at time t.

At each time point, Z weight is removed in most profitable order as in Equation (3.2), i.e.

$$y_{t+1} = y_t + \Delta y_t - Z$$

Of the transactions remaining, $y_t^{>B} = y_{t,\mathrm{FR}>\mathrm{FR}(\tau^B)}$ represents the portion of transactions which would prevent τ^B from confirming, and

$$\sum_i \omega(\tau_{t,i}^A) + y_t^{>B} > Z \quad (\mathrm{FR}(\tau_{t,i}^A) = \mathrm{FR}(\tau^B) + \alpha;\ \alpha > 0) \tag{3.3}$$

must hold for each time point t.

For each time point $t > 0$, fee estimators will converge on estimations $\mathrm{FR} > \mathrm{FR}(\tau_{t-1,i}^A) > \mathrm{FR}(\tau^B)$ and the weight requirement (first component of Equation (3.3)) will drop in response to a growth in $y_t^{>B}$ (second component). In other words, it would become cheaper each block to spam the network for A, assuming enough transactors were willing to pay $\mathrm{FR} > \mathrm{FR}(\tau^B)$.

Because $y_t^{>B}$ grows as $\mathrm{FR}(\tau^B)$ shrinks, the viability increases proportionately to the initial claim transaction fee rate.

The total fees f^A paid by A in order to gain d bitcoin becomes

$$f^A = \sum_{i=0}^{t} \sum_j \omega(\tau_{i,j}^A) \cdot \mathrm{FR}(\tau_{i,j}^A)$$

The attack is profitable for A as long as $f^A < d$, after which A begins to lose money. However, it will always be beneficial for A if τ^A manages to make

it into a block, as the alternative is that A ends up with a balance $-f^A$, instead of $c - f^A$.

It is safe to assume that B would use a higher than average fee rate, as they know that A is attempting to steal the funds, so this attack may only be viable if the fee rate experiences a jump soon after τ^B is broadcast. This depends on how much transactors are willing to spend during this high-volume period. The more transactors willing to pay higher than $\mathrm{FR}(\tau^B)$, the cheaper the attack becomes.

Replace-By-Fee (RBF)

With RBF, τ^B becomes τ^B_t, and B can now bump $\mathrm{FR}(\tau^B_t)$ as appropriate, but this functionality must be present in Lightning node implementations, and the initial transaction τ^B_0 must have RBF enabled. By observing the mempool, it is possible to respond almost instantly to where[15]

$$\mathrm{FR}(\tau^B_t) = \mathrm{FR}(\tau^A_{t,i}) + \beta \quad (\beta > 0). \tag{3.4}$$

This would require A to RBF-bump $\tau^A_{t,i}$ in response, so that they again satisfy the requirement in Equation (3.3). Each time B does this, there is a chance that τ^B_t is mined before all of $\tau^A_{t,i}$ have propagated, which would result in an immediate failure for A.

In addition, the cost increase for B is

$$\omega(\tau^B_t) \cdot \mathrm{FR}(\tau^B_t) - \omega(\tau^B_{t-1}) \cdot \mathrm{FR}(\tau^B_{t-1}) \approx (\alpha + \beta)\omega(\tau^B_t) \tag{3.5}$$

whereas the cost increase for A is

$$\begin{aligned} \sum_i \omega(\tau^A_{t,i}) \cdot \mathrm{FR}(\tau^A_{t,i}) - \sum_j \omega(\tau^A_{t-1,j}) \cdot \mathrm{FR}(\tau^A_{t-1,j}) & \\ \approx (\alpha + \beta) \sum_i \omega(\tau^A_{t,i}) & \\ \approx (\alpha + \beta)(Z - y_t^{>B}) & \end{aligned} \tag{3.6}$$

which, at the worst case (where $y_t^{>B} = 0$), is $(\alpha + \beta)Z$. Each time B RBF-bumps, $y_t^{>B}$ becomes smaller, so it follows that A ends up paying significantly more than B in fees each time B RBF-bumps. In fact, the deterrent D for A to continue their attack rises in proportion to the required additional fee in order to outbid B (Equation (3.7)).

$$D = \frac{\beta(Z - y_t^{>B})}{c} \tag{3.7}$$

As such, B should attempt to fit β so that it (1) minimizes the direct fee cost for B and (2) minimizes the incentive for A to continue the attack.

[15] The t notation is slightly abused here and in following paragraphs, to both mean "block at point t" and to mean "the current (for t) or previous (for $t-1$) fee rate", even though the latter case could repeat within the span of a single block.

$$\beta = \frac{Dc}{Z - y_t^{>B}} \tag{3.8}$$

If the deterrent $D = 1$, we get that $\beta = \frac{c}{Z - y_t^{>B}}$, which would mean that A had to pay the entire funding amount c in order to out-bump τ^B. This would obviously mean B paid a very high fee, but it would basically guarantee that A stopped their attack.

If the block weight capacity (Z) increases, the attack becomes proportionately more expensive to execute, depending on transaction throughput (y_t) and fee rate acceptability among transactors ($\frac{y_t^{>B}}{y_t}$). While the losses incurred to B due to RBF-bumping τ^B will be reduced significantly in proportion to the costs incurred to A, proper implementation of the above response mechanism in layer 2 tech software should be sufficient to deter any attempts at performing this kind of attack.

Given this, this attack is not considered a cause for concern, but this is under the assumption that Lightning nodes and other layer 2 technologies with similar mechanics implement RBF bumping and auto-detection of, as well as proper RBF-bumping in response to, spam attacks as the one mentioned above.

Note that there is a small window between the broadcasting of τ_0 and the time that it is mined into a block, in which B can broadcast τ^B. Most mining nodes will group these together into a "packet", which means either τ_0 and τ^B are mined together, or none of them are. Since τ_0 cannot be modified, A cannot RBF-bump it, even if RBF was enabled, and would thus have lost immediately. A may need to make use of a mining node paid out of band to ensure B does not see τ_0 before it is mined.

Note also that B can pay one or several miners out of band to include τ^B in their next block, effectively circumventing the spam attack. This relies on benevolent actors, and relying on this method would negatively impact decentralization.

Finally, note that this attack works for cross-chain atomic swaps as well (the larger the amount the better), where the attackers reclaim their paid coins on the sending chain by preventing the payee from claiming it within the time lock period. In fact, this is arguably more viable than the Lightning Network case above.

3.1.4 Distributed attack on time-locked contracts

The attack mentioned in 3.1.3 can be extended to multiple attackers $A^0, A^1, \cdots, A^N$ (some of which may be the same person with multiple channels) which are performing a coordinated attempt to overwhelm the mempool for ℓ blocks, for an aggregate profit of $\sum_k (d_k - f_k^A)$.

- The attackers simultaneously broadcast $\tau_{0,0}, \cdots, \tau_{N,0}$, potentially by paying a miner out of band to include them in their next block without revealing them beforehand to $B^0, \cdots, B^N$.

- The attackers spam the mempool in coordination, preventing τ_k^B from confirming.

The benefit of this approach is that each attacker need only spend

$$\frac{1}{N}(Z - y_t^{>B}) \tag{3.9}$$

worth of weight. The attackers would most likely agree to push out *all* claim transactions τ_k^B, which means the attackers would need a fee rate greater than $\max(\tau_k^B)$, which means some attackers would pay an unnecessarily high fee rate compared to their required rate, in order to aid their accomplices.

To mitigate this in part, the attackers might scale the responsibility based on the profit d_k, so that Equation (3.9) becomes

$$\frac{d_k}{\sum_{i=0}^{N} d_i}(Z - y_t^{>B}) \tag{3.10}$$

i.e. so that attackers with a smaller profit would pay less in fees compared to attackers with a large profit. This would incentivize attackers to have approximately similar profits.

This "distributed" variant of the attack is much more powerful than the single case described in 3.1.3. Its weakness lies in the incentive to cooperate by collectively bumping out the highest fee rate transaction in the group. There is potential to cause the attackers a loss in profit by identifying the attack and the participants, and to greatly bump the most profitable transaction so high that other participants would lose money by trying to out-bump it, without immediately RBF-bumping the other claim transactions.

The counter would be to "cut the losses" and abandon that transaction but to attempt to fill the mempool for the remaining ones. Because the attackers have now dropped one participant, the remaining attackers would need to cover the mempool for the remaining transactions, increasing the overall cost of each participant. The operation could be repeated, until the attackers give up, or until all the transactions have been bumped so high neither of them are profitable. If the defenders have accurately estimated the number of participants, each loss-cutting transaction could be estimated fairly precisely via Equation (3.7). This all works because of the imbalance expressed in Equations (3.5) and (3.6), and the fact the cost of bumping for the attackers increases with each dropped participant. It is important, however, that the fees are not gradually increased, because whatever fees the attackers pay to uphold the attack are permanently lost; each attacker will always want τ_k^A to confirm, as it means a direct c_k increase in bitcoin, even if at a loss, so the closer the fee required to bump at a given time is to c_k, the less incentivized the attacker will be to continue the attack (see Equation (3.7)).

The drawback for B is obvious. They will lose a significant amount of their funds due to the fee race. In particular the "sacrificial" transaction mentioned above would cause a potentially big loss for the owner.

The merit of this approach is that it acts as a deterrent to attempt the above kind of distributed attack, but it relies on the software being able to detect the attack and coordinate an appropriate response, although this should probably not be automatic[16]. More importantly, it relies on benevolent actors willing to lose funds to deter the attackers from even trying. Some form of reward system may assist in this, although this is outside the scope of this report.

An increase in block weight capacity would increase the cost of the attack and weaken its viability. However, this could easily be countered by simply having a decent network for finding accomplices, and to perform the attack on a larger (higher N) scale.

Note again that one or several of the individual participants B^k can pay one or several miners out of band to include τ_k^B in their next block, protecting themselves. Again, this relies on benevolent actors, and relying on this method would negatively impact decentralization.

[16]The response would be to *not* attempt to RBF-beat the attackers until the sacrifice is made, not to make an arbitrary sacrifice, as the loss would be quite large if the software misjudged the situation.

CHAPTER

FOUR

FEE RATE OPTIMIZATION VIA THE MEMPOOL STATE

This chapter expands on the concept of using RBF (replace-by-fee) in wallet software in a more automatic and user friendly way, where users set a maximum fee, rather than an explicit one (4.1), then goes into detail on how to combine the current fee rate estimators which use block statistics, with a fee rate estimator using the mempool (4.2).

4.1 Automatic transaction bumping via RBF

A problem with using the mempool to estimate fees is that you can never be sure when the next block is going to be mined. If it takes less than the target of 10 minutes, you may over-estimate, and if it takes longer, you may under-estimate, simply because transactions attempt to out-bid each other.

A natural extension to enabling RBF by default would be for wallet software to periodically recheck the mempool and bump the transaction fee automatically, which would address both of the issues mentioned above. A drawback of this is that the wallet software needs to remain active even after sending the transaction, but it may be sufficient for it to do the recheck-and-bump whenever it is brought to the foreground, as users will most likely bring the application up to check on the transaction, especially if it is urgent enough to warrant fine-tuning.

$$\textsc{ShouldBump}(T, t, Z) = tZ < y_{\mathrm{FR} \geq \mathrm{FR}(T)} \tag{4.1}$$

For instance, if a user wants a transaction to confirm within the next t blocks ("target"), the wallet software may choose to bump the fee if the accumulated weight of all transactions above itself in the mempool exceed t blocks worth of weight (Equation (4.1), where T is the transaction being bumped, t is the target (in blocks to confirm), and Z is the block size).

Using Equation (4.1) it is trivial to derive a method for finding an optimally minimal fee bump (Equation (4.2), where T_i represents a bumped form of the transaction T where the fee has been increased by i).

$$\begin{aligned} &\textsc{FeeDelta}(T, t, Z, x) = \min(i) \\ &\text{provided} \quad \neg\textsc{ShouldBump}(T'_i, t, Z) \vee \mathrm{FR}(T'_i) \geq x \\ &\text{where} \quad T'_i = \{T; \mathrm{FR}(T'_i) = \mathrm{FR}(T) + i\} \end{aligned} \tag{4.2}$$

A wallet would make periodic checks at appropriate times by calling FeeDelta on each unconfirmed transaction; if the resulting value is greater than 0, the transaction is bumped. Since the user has given a maximum acceptable fee, the transaction will never exceed acceptable values, no matter what happens (unless the user increases the maximum).

Note that even a full node will not see the entire mempool, but a subset of it. This affects the accuracy of FeeDelta and ShouldBump proportionately.

4.2 Combining fee rate estimators

In reality, at least in the case of Bitcoin Core, wallets do not use the mempool state *at all* when estimating fees, contrary to this report which has *only* been using the mempool up until this point. Instead, most wallets look at past blocks and use statistics to determine a fairly reliable fee rate that would put the transaction in the next block. By using the mempool and "pretending" to make a block from the observable transactions, a wallet can get a rough representation of what the next block will look like, depending on how much time passes until the next block is found.

This assumes that miners are rational and open, but there is no basis for such an assumption. It is highly probable that miners take out-of-band payments to include specific transactions, which would bypass the mempool market.

The mempool state for a given node is just a local state of the actual mempool, sort of like an approximation of what everyone else is seeing. There is some delay in transaction relay, and as such, the mempool is not "perfect knowledge".

The mempool can be manipulated, and is by nature highly volatile[17]. Using only the mempool to estimate fees is as such not recommended. Instead, the fee estimation from the mempool state should be given as a lower boundary to the regular fee estimation function(s) used, as in Equation (4.3).

$$f(\cdot) = \min(f_{\textsc{MPOpt}}(\cdot), f_{\textsc{SmartEst}}(\cdot)) \tag{4.3}$$

In Bitcoin Core, the fee rate estimation (here called $f_{\textsc{SmartEst}}(\cdot)$) was improved[18,19] in version 0.15. It currently uses three horizons ($\frac{t}{2}$, t, and $2t$, where

[17] See e.g. https://www.reddit.com/r/Bitcoin/comments/764nt7/be_warned_ledger_nano_s_transaction_accelerator/?st=J8QFCAAU&sh=3e5e642a in which a user is recommended a $7k fee by his wallet due to mempool optimizations gone awry.

[18] https://github.com/bitcoin/bitcoin/pull/10199

[19] https://gist.github.com/morcos/d3637f015bc4e607e1fd10d8351e9f41

t is the number of blocks until the transaction should be confirmed), each with a different success rate (60%, 85%, and 95% respectively).

Each of these estimations is based on fee rate buckets, which are used to keep track of when transactions in a given fee rate range were mined. There are two modes; conservative, which has an additional requirement of 95% success rate over a longer time horizon, and economic, which is relieved of this requirement.

This works well under normal circumstances, but tends to result in overly high estimations when the transaction throughput drops rapidly or miners are unusually lucky in finding multiple blocks within a short time span, i.e. when the last couple of blocks have relatively high fees, but the mempool is relatively empty.

Using Equation (4.3) would in many cases dampen this effect.

4.3 Results

Simulations were run based on a simple algorithm for determining the mempool-based fee rate. These were applied according to Equation (4.3), which prevented mempool volatility from causing a higher loss than the current smart fee estimator of Bitcoin Core.

The algorithm used to determine the mempool fee begins by selecting a point inside an imaginary block, if one was mined based on the local mempool at that point.

$$P = \max(0.10, (0.10 + Q + C) \times (1 - 0.05T)) \tag{4.4}$$

Q is 0.03 if it's been less than 170 seconds since the last block was discovered, and 0 if it's been more; C is 0.05 if the estimation is *conservative* (i.e. not *economic*); T is the confirmation target in number of blocks.

In other words, start with 10%; if the previous block was found within 2m10s, add 3%; if the estimation is conservative, add another 5%; multiply the result by $1 - 0.05$(confirmation target), i.e. 0.95 for 1, 0.90 for 2, and so on. If the result is lower than 10%, pick 10%.

Next, the algorithm creates a simulated block based on the mempool, which is simply done by adding transactions in descending (most profitable first) fee rate order until the block is full.[20]

The algorithm takes the list of transactions $\vec{\tau}^*$ in the simulated block in ascending fee rate order (the opposite of what was used to create the block above), and generates a fee rate estimation.

$$\mathrm{FR}^*(\vec{\tau}^*, P) = \mathrm{FR}(\tau^*_x) \quad x = |\vec{\tau}^*|P \tag{4.5}$$

This would put the transaction in the simulated block at the x^{th} position from the bottom.

[20] Note that the implementation used disregards transaction dependencies, such as where τ_1 is used as input for τ_2, and both are in the mempool still. The algorithm would happily include τ_2 in the simulated block, even though τ_1 is needed.

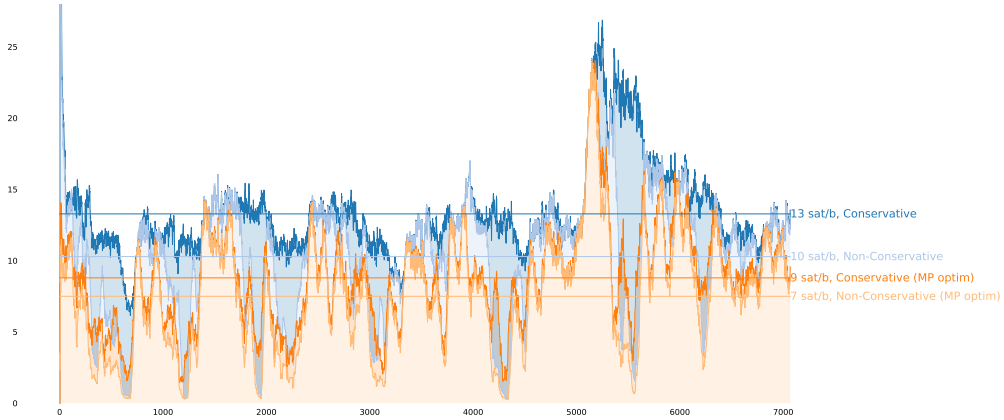

Figure 4.1: Overpayment in sat/b for each mode. As can be noted, the MP optimized mode reacts much more quickly to mempool state changes, where the non-optimized variants have a delayed reaction. Overall, the non-conservative option saves 30%, and the conservative option saves 31% in fees.

The simulation was run over the period August through October of 2017, with approximately 100,000 estimation attempts made for each type.

Figure 4.1 shows the average overpayment (where 0 means the transaction went in exactly at the bottom of the next block) for 1-target transactions.

Figures 4.2 – 4.5 show number of transactions that were not confirmed after the given number of blocks. After 1 block (Figure 4.2), there is a fair amount of error (3.01% optimized vs 1.63% unoptimized for economic modes), but most of these under-estimated transactions ended up being confirmed a few blocks later (Figure 4.4). After 10 blocks, 11 unconfirmed transactions remained (Figure 4.5), most (10) of them stemming from the non-conservative estimator.

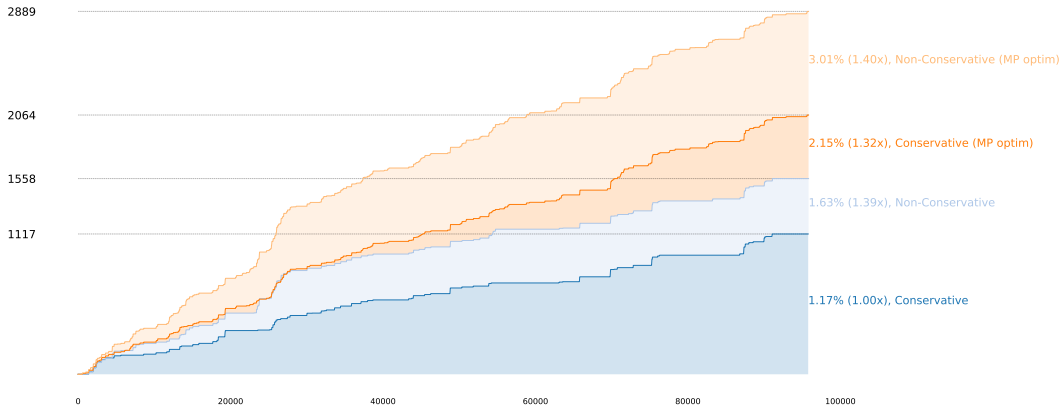

Figure 4.2: Unconfirmed transactions after 1 block over total transactions. The Bitcoin Core 0.15 "conservative" estimation method ends up with 1117 (1.17%) transactions not confirming, with the "non-conservative" (or "economic") mode getting 1558 (1.63%). The conservative method using mempool optimization has 2064 unconfirmed (2.15%), and the non-conservative mempool optimized has 2889 (3.01%).

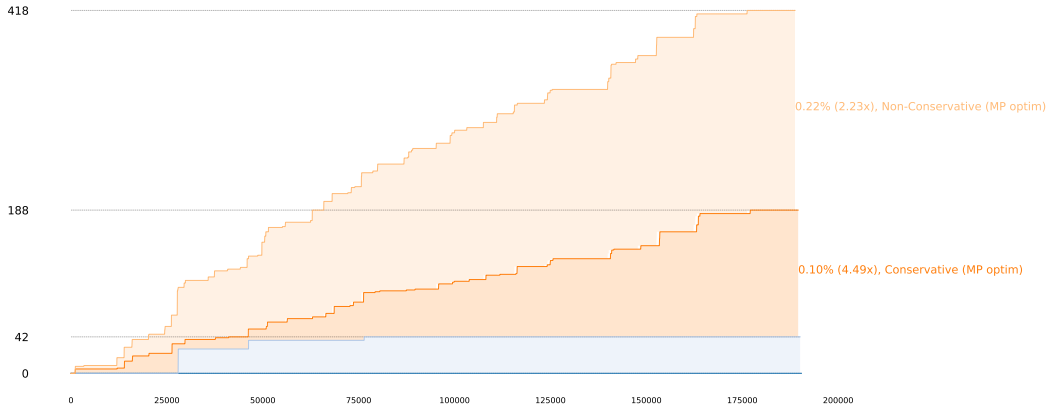

Figure 4.3: Unconfirmed transactions after 2 blocks over total transactions. Here, none of the conservative remain unconfirmed, and 42 non-conservative remain; 188 conservative/optimized, and 418 non-conservative/optimized.

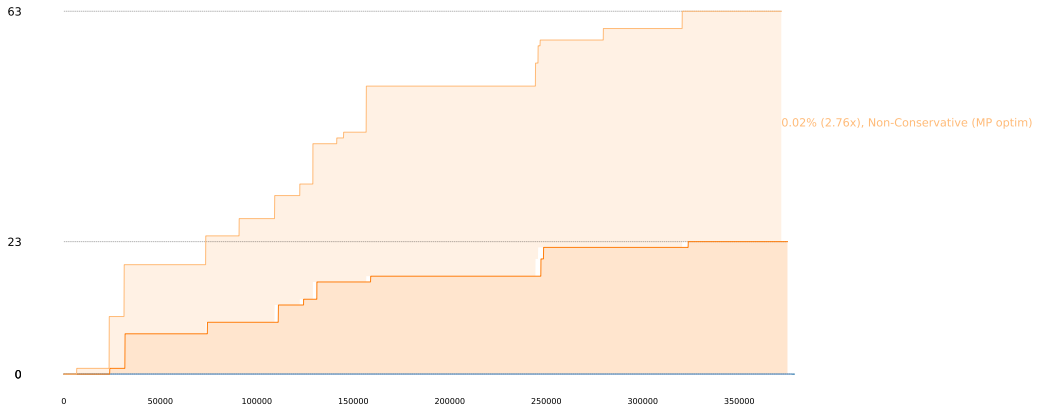

Figure 4.4: Unconfirmed transactions after 4 blocks over total transactions.

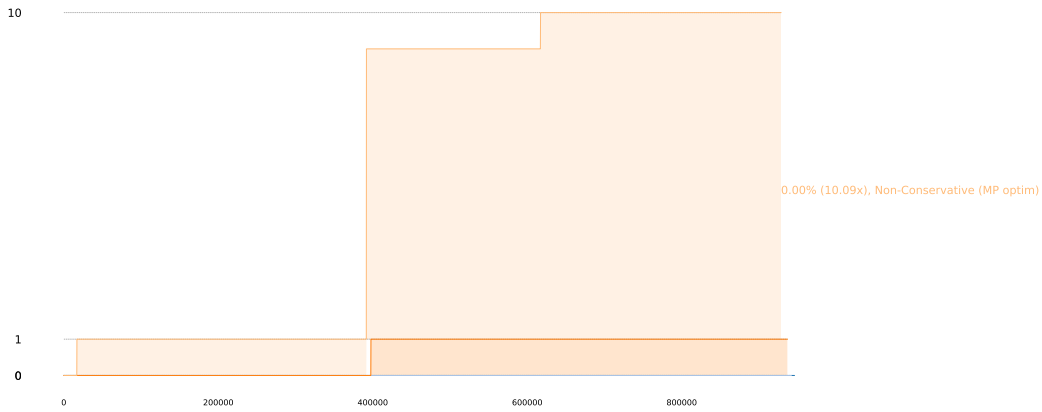

Figure 4.5: Unconfirmed transactions after 10 blocks over total transactions. 1 conservative and 10 non-conservative optimized transactions remain unconfirmed. The remaining transactions have all been confirmed.

CHAPTER

FIVE

CONCLUSIONS

The assumption that all transactions are mined at all times leads to a number of problems:

- We end up with all transactions using the minimum fee rate[21]. At the current price levels, miners would get $55 USD per MB of block size per block.
- To retain the equilibrium in terms of bitcoin per block (e.g 10 bitcoin per block), we would need 30 MB blocks by 2020 and 80 MB blocks when subsidy halving is completely replaced by fees. As the price of bitcoin rises, the minimum fee rate would decrease, and as a result the block size requirement would increase; at $1 mln USD/BTC, each block would take 14 GB. Even the most optimistic (and compact) 30 MB size would severely increase centralization and decrease stability and accessibility to the Bitcoin network.
- To retain the equilibrium in terms of miner profit (roughly $65,000 USD per block), we would need 77 MB blocks at the next halving if users paid 10 cents per transaction, although this would drop with the rising price of bitcoin, e.g. to 7.2 MB for $10k USD/BTC. This drop is tied to the subsidy halving, however, and would ultimately fall off completely, leaving blocks at 186 MB, regardless of USD/BTC price. If miners required a higher profit, the block size would increase proportionately; e.g. at a $100k USD/block profit, the block size would be nearly 300 MB.
- Arguably most problematic of all is, for the case with a low enough subsidy, if there are not enough transactions in the mempool at the time, there is no reason for miners to find a new block, and the blockchain will

[21] There is no reason for users to pick a fee higher than the minimum, if there is no risk of the transaction being "bumped" by a higher-paying transaction. For sufficiently large blocks, everybody ends up paying the minimum fees until such a time as the transaction throughput beginning to exceed the block size.

effectively grind to a halt until enough value has accumulated in the mempool. This results not only in higher centralization and unpredictability, but also renders the premise of "most work secures the chain" ineffective, because miners would now profit from remining the chain tip block(s) while waiting, and it would decrease the difficulty, lowering the overall security of the chain.

Removing this assumption alleviates most of these issues, but results in a competitive mempool market with "winners" and "losers":

- Transactions will approach a fee rate individually defined by their respective creators. The miner profit would vary, depending on mempool size, fee rate estimator precision and accuracy, and value of transacting. Users would weigh speed to be confirmed vs cost, just as they are now.
- The bitcoins per block received by miners would reach an equilibrium related to the aggregate value of transacting, as well as the transaction throughput. It would not directly satisfy miner profit requirements, but it would be far better than "minimum fee rate $\times$ weight".
- The mempool would very seldomly be empty, if ever, removing the problem with miners "pausing" their equipment due to a lack of profitable transactions once the subsidy has become a less significant part of the miner profits.

A full mempool results in a number of issues, some of which can be alleviated to a certain extent by optimizing fee rate estimators, and educating users on how fees work, e.g. via improved UI elements in wallet software.

A high fee rate would lead to a decrease in adoption rate, but with layer 2 technology like Lightning Network, higher fees would be more acceptable, as they go from per-transaction fees to per-charge fees over many transactions.

CHAPTER

SIX

ACKNOWLEDGEMENTS

Thank you to DG Lab for providing the funding behind this paper.

Special thanks to Nicolas Dorier, Anditto Heristyo, and Max Justicz for feedback on content and language.

APPENDIX

A

DERIVATION OF FEE FUNCTION GIVEN IN EQUATION 2.3

From Equation (2.3),

$$f(y, Z) = \frac{1}{4} 1.1^{\lfloor \frac{y}{0.95Z} \rfloor} \text{ satoshi}/\omega$$

- When a user creates a new transaction, they look at the mempool and finds the fee that would put their transaction at the bottom 5% point to minimize their cost, with a minimum 10% increase in fees compared to the closest competing transaction at the threshold.
- If no transaction exists ($y = 0$) or if the mempool is smaller than 95% of a block ($y < 0.95Z$), the transaction uses a 1 satoshi per byte ($\frac{1}{4}$ satoshi per weight unit) fee (the minimum required to be relayed by nodes, at this point in time).
- The above condition applies for the first $0.95Z$ weight units, after which the user will pick a 10% higher fee than the competing transaction (which is initially at 1.0 sat/b), i.e. $1.0 \cdot 1.1 = 1.1$ ($y = 0.95Z$).
- After an additional $0.95Z$ weight units ($y = 2 \cdot 0.95Z$), we have filled the available 95% with $\frac{1}{4}1.1$ fee rate transactions, and as such, a new transactor will pick a 10% higher fee, i.e. $\frac{1}{4}1.1 \cdot 1.1 = \frac{1}{4}1.1^2$.
- We apply induction to derive the generalized $f(y, Z) = \frac{1}{4} 1.1^{\lfloor \frac{y}{0.95Z} \rfloor}$.

APPENDIX

B

IMPACT OF SUBSIDY HALVING

Looking at miner revenue per block for the last year[22], the revenue is around 14 bitcoin per block, with peaks around 16.5 and has a lowest point around 12.8.

Table B.1 shows how big a portion of miner fees are made up by the subsidy. It assumes that the miners will receive the same amount of bitcoin per block, i.e. 14 bitcoin. While this assumption is *most likely incorrect*, it provides us with an example of how things will evolve, if the miner revenue remains somewhat stable. The table shows that less than 11% of miner fees are from transaction fees. In the next subsidy halving in three years in 2020, this will jump by roughly 45%. At the next halving in 2024, it will be over 77%, and the subsidy will make up less than 10% come 2032.

Table B.2 removes the assumption of bitcoin revenue and instead assumes miners receive approximately $65,000 USD per block instead, and also assumes that fees make up 11% of the total revenue (with a minimum cap of total profit set to 0.01 BTC). Again, it must be noted that these assumptions are arbitrary, and only serve to give an idea of how things would evolve under the given parameters. The bitcoin price that must follow for this to apply is listed as well in the right-most column. The lower limit and the price requirement ends up capping the bitcoin price at $6.5 million USD. In this scenario, the user fees still comprise over 50% of the miner revenue, although this begins at a later date (2064, 47 years in the future); this is based on the 0.01 BTC minimum assumption, though, which if adjusted will move the point in time at which this takes effect. E.g. with a 1 bitcoin minimum assumption, the fees would comprise over 50% of the miner revenue from the 2036 halving in 19 years.

Table B.3 shows the scenario where the fees grow by 5% each subsidy halving, and how this would affect e.g. the bitcoin price and revenue in terms of bitcoin. Again note that the $65,000 USD per block and 5% growth assumptions are arbitrary. The price peaks, and the revenue hits its lowest point, in 2040, at which point the 5% increase in fees outpaces the revenue lost from subsidy halving.

[22] https://www.smartbit.com.au/charts/miner-revenue-per-block

Subsidy	Year	Bitcoin in fees	% Fees
12.50000000	2016	1.50000000	10.71429%
6.25000000	2020	7.75000000	55.35714%
3.12500000	2024	10.87500000	77.67857%
1.56250000	2028	12.43750000	88.83929%
.78125000	2032	13.21875000	94.41964%
.39062500	2036	13.60937500	97.20982%
.19531250	2040	13.80468750	98.60491%
.09765625	2044	13.90234375	99.30246%
.04882813	2048	13.95117188	99.65123%
.02441406	2052	13.97558594	99.82561%
.01220703	2056	13.98779297	99.91281%
.00610352	2060	13.99389648	99.95640%
.00305176	2064	13.99694824	99.97820%
.00152588	2068	13.99847412	99.98910%
.00076294	2072	13.99923706	99.99455%
.00038147	2076	13.99961853	99.99728%
.00019073	2080	13.99980927	99.99864%
.00009537	2084	13.99990463	99.99932%
.00004768	2088	13.99995232	99.99966%
.00002384	2092	13.99997616	99.99983%
.00001192	2096	13.99998808	99.99991%
.00000596	2100	13.99999404	99.99996%
.00000298	2104	13.99999702	99.99998%
.00000149	2108	13.99999851	99.99999%
.00000075	2112	13.99999925	99.99999%
.00000037	2116	13.99999963	100.00000%
.00000019	2120	13.99999981	100.00000%
.00000009	2124	13.99999991	100.00000%
.00000005	2128	13.99999995	100.00000%
.00000002	2132	13.99999998	100.00000%
.00000001	2136	13.99999999	100.00000%
.00000001	2140	13.99999999	100.00000%
.00000000	2144	14.00000000	100.00000%

Table B.1: Portion of miner profit from subsidy vs fees, assuming a steady revenue of 14 bitcoin per block.

Subsidy	Year	Fees	Revenue	% Fees	BTC price
12.50000000	2016	1.54494375	14.04494375	11.00000%	$4,628
6.25000000	2020	.77247188	7.02247188	11.00000%	$9,256
3.12500000	2024	.38623594	3.51123594	11.00000%	$18,512
1.56250000	2028	.19311797	1.75561797	11.00000%	$37,024
.78125000	2032	.09655898	.87780898	11.00000%	$74,048
.39062500	2036	.04827949	.43890449	11.00000%	$148,096
.19531250	2040	.02413975	.21945225	11.00000%	$296,192
.09765625	2044	.01206987	.10972612	11.00000%	$592,384
.04882813	2048	.00603494	.05486306	11.00000%	$1,184,768
.02441406	2052	.00301747	.02743153	11.00000%	$2,369,536
.01220703	2056	.00150873	.01371577	11.00000%	$4,739,072
.00610352	2060	.00389648	.01000000	38.96484%	$6,500,000
.00305176	2064	.00694824	.01000000	69.48242%	$6,500,000
.00152588	2068	.00847412	.01000000	84.74121%	$6,500,000
.00076294	2072	.00923706	.01000000	92.37061%	$6,500,000
.00038147	2076	.00961853	.01000000	96.18530%	$6,500,000
.00019073	2080	.00980927	.01000000	98.09265%	$6,500,000
.00009537	2084	.00990463	.01000000	99.04633%	$6,500,000
.00004768	2088	.00995232	.01000000	99.52316%	$6,500,000
.00002384	2092	.00997616	.01000000	99.76158%	$6,500,000
.00001192	2096	.00998808	.01000000	99.88079%	$6,500,000
.00000596	2100	.00999404	.01000000	99.94040%	$6,500,000
.00000298	2104	.00999702	.01000000	99.97020%	$6,500,000
.00000149	2108	.00999851	.01000000	99.98510%	$6,500,000
.00000075	2112	.00999925	.01000000	99.99255%	$6,500,000
.00000037	2116	.00999963	.01000000	99.99627%	$6,500,000
.00000019	2120	.00999981	.01000000	99.99814%	$6,500,000
.00000009	2124	.00999991	.01000000	99.99907%	$6,500,000
.00000005	2128	.00999995	.01000000	99.99953%	$6,500,000
.00000002	2132	.00999998	.01000000	99.99977%	$6,500,000
.00000001	2136	.00999999	.01000000	99.99988%	$6,500,000
.00000001	2140	.00999999	.01000000	99.99994%	$6,500,000
.00000000	2144	.01000000	.01000000	100.00000%	$6,500,000

Table B.2: Bitcoin fees and price given a set revenue of $65,000 USD per block, where fees are locked at 11%, with a minimum revenue of 0.01 BTC (which is taken from fees, which is why they begin to rise in 2060).

Subsidy	Year	Fees	Revenue	% Fees	BTC price
12.50000000	2016	1.54494375	14.04494375	11.00000%	$4,628.00
6.25000000	2020	1.62219094	7.87219094	20.60660%	$8,256.91
3.12500000	2024	1.70330048	4.82830048	35.27743%	$13,462.29
1.56250000	2028	1.78846551	3.35096551	53.37165%	$19,397.39
.78125000	2032	1.87788878	2.65913878	70.62019%	$24,444.00
.39062500	2036	1.97178322	2.36240822	83.46497%	$27,514.30
.19531250	2040	2.07037238	2.26568488	91.37954%	$28,688.90
.09765625	2044	2.17389100	2.27154725	95.70089%	$28,614.86
.04882813	2048	2.28258555	2.33141368	97.90564%	$27,880.08
.02441406	2052	2.39671483	2.42112889	98.99162%	$26,846.98
.01220703	2056	2.51655057	2.52875760	99.51727%	$25,704.32
.00610352	2060	2.64237810	2.64848162	99.76955%	$24,542.36
.00305176	2064	2.77449701	2.77754876	99.89013%	$23,401.93
.00152588	2068	2.91322186	2.91474774	99.94765%	$22,300.39
.00076294	2072	3.05888295	3.05964589	99.97506%	$21,244.29
.00038147	2076	3.21182710	3.21220857	99.98812%	$20,235.30
.00019073	2080	3.37241845	3.37260919	99.99434%	$19,272.91
.00009537	2084	3.54103937	3.54113474	99.99731%	$18,355.70
.00004768	2088	3.71809134	3.71813903	99.99872%	$17,481.86
.00002384	2092	3.90399591	3.90401975	99.99939%	$16,649.51
.00001192	2096	4.09919571	4.09920763	99.99971%	$15,856.72
.00000596	2100	4.30415549	4.30416145	99.99986%	$15,101.66
.00000298	2104	4.51936327	4.51936625	99.99993%	$14,382.55
.00000149	2108	4.74533143	4.74533292	99.99997%	$13,697.67
.00000075	2112	4.98259800	4.98259875	99.99999%	$13,045.40
.00000037	2116	5.23172790	5.23172827	99.99999%	$12,424.19
.00000019	2120	5.49331430	5.49331448	100.00000%	$11,832.56
.00000009	2124	5.76798001	5.76798010	100.00000%	$11,269.11
.00000005	2128	6.05637901	6.05637906	100.00000%	$10,732.49
.00000002	2132	6.35919796	6.35919799	100.00000%	$10,221.41
.00000001	2136	6.67715786	6.67715787	100.00000%	$9,734.68
.00000001	2140	7.01101575	7.01101576	100.00000%	$9,271.12
.00000000	2144	7.36156654	7.36156654	100.00000%	$8,829.64

Table B.3: Bitcoin fees and price given a set revenue of $65,000 USD per block, where fees grow by 5% each subsidy halving.