Block Publication Incentives For Miners

Jul 29, 2016

Thanks goes to Jonathan HB for reviewing the math in an early draft; all mistakes still mine!

If I'm a miner and I find a block, why should I publish that block to the world?

It's a simple question, yet it turns out to be at the heart of Bitcoin's scalability problem, and security model. Here we'll take a high-level, theoretical, look at what incentives exist in the Bitcoin protocol to publish blocks, and what those publication incentives mean for decentralization.

Contents

1. Decentralization Goals
2. Why Publish At All?
    2.1. Majority Hashing Power
    2.2. Minority Hashing Power
3. Delaying Block Publication
    3.1. Temporary Block Withholding
    3.2. Non-Negligible Propagation Delays
        3.2.1. Small Miners
        3.2.2. Expected Return Ratio
    3.3. Non-Uniform Propagation Delays
        3.3.1. Slow Blocks
4. Footnotes

1 Decentralization Goals

We'll discuss block publication incentives in terms of fairness between different classes of miners; by fairness we mean the profit per unit hashing power. In an ideal world

miners/pools[1] with a small percentage of total hashing power will have the same per-unit revenue as miners/pools with a large percentage of total hashing power. If that is not true, larger miners are likely to get even bigger as they reinvest their profits, and hashers are incentivised to move their hashing power to larger pools over smaller ones, both centralization pressures.

In short, we want to design a protocol that incentivises what we want: a diverse mining community without single points of failure. Of course, due to overheads like running a full node mining will never be entirely fair but we'd like to limit that unfairness as much as possible.

2 Why Publish At All?

The most basic incentive for miners to publish blocks is to get paid: assuming buyers only accept coins from blocks that their full nodes have validated, miners are forced to publish their blocks to bitcoin buyers to get paid for the blocks they mine. Equally, with current blocksizes the cost to publish those blocks widely is low enough that we can rely on altruistically run P2P nodes.

However, this isn't by itself a direct incentive to publish blocks to other miners; publish blocks to bitcoin buyers only needs to be done occasionally when a miner sells their reward. Why should miners publish to other miners?

2.1 Majority Hashing Power

For a short-term rational miner with a majority of hashing power there isn't a clear incentive: they have the majority, so if they choose not to publish the chain they're working to other miners it always will eventually have more work than the chain their competition is working on. When they publish their chain as part of selling their coins their competitions' chain is simply wiped out, eventually pushing them out of business and allowing the majority miner to save money by turning off some of their hashing power; if users complain about reorgs the majority miner can simply publish more frequently, every time they are in the lead.

Of course, such blatantly unfair behavior has a second order effect: the fact that no-one else can mine is strong evidence that Bitcoin is now highly centralized, and a rather dubious investment. This *should* significantly reduce the exchange rate, but as we'll see later such attacks aren't necessarily easily detected, or even clearly attacks.

2.2 Minority Hashing Power

For a minority miner, if other miners don't build on top of their blocks they won't be a part of the most-work chain. So if they don't publish, their competition will create a longer chain in the long run, and when they wish to sell their coins buyers will consider those coins invalid.

However, note how that incentive presumes that other miners need a block to build on top of it - that's not clearly true. At minimum, a valid empty block can be created by a miner knowing only the previous block's hash, height, and median time, so long as the previous block is valid. Equally if the miner can learn what modifications to make to the UTXO set - perhaps with a segwit block sans witnesses - they can also collect transaction fees.

3 Delaying Block Publication

While a minority miner needs to publish their blocks to other miners at some point, that doesn't necessarily mean a minority miner has an incentive to publish in a *timely* fashion. If we publish our blocks immediately, we maximize our chances of our block ending up in the final consensus chain (minimizing our stale rate[2]).

However it turns out that in many circumstances deliberately delaying publication is to our advantage: while delaying increases our stale rate, it increases our competitors' stale rates even more, resulting in us finding more blocks than them. In the short run this potentially increases our transaction fee revenue; in the medium term (one to two weeks[3]) difficulty adjusts downwards as the stale blocks are taken into account by the difficulty adjustment leaving us ahead on fees, and no worse off for subsidy revenue; in the long run we benefit if any of our competition is forced out of business.

This strategy was first discovered and initially analyzed by the Bitcoin development community in 2010[4]. It was later rediscovered[5] and more rigorously analysed by Eyal and Sirer[6], who gave their particular take on it the name "Selfish Mining".

For the sake of simplicity we'll look at a somewhat simplified analysis originally published[7] by myself, which gets very similar results to Sirer's more rigorous work modulo the (as we'll see later, often unnecessary) active network attack they propose.

3.1 Temporary Block Withholding

Suppose we're a miner with Q hashing power, and the rest of the network has $1 - Q$

hashing power. We'll assume propagation time is negligible; between the time a miner publishes a block, and the time all other miners receive the block, the chance of another miner finding a block is essentially zero. If we find a block and do *not* propagate it immediately there are three possible outcomes, with the following probabilities:

1. Q — We find the next block. We're now two blocks ahead, so if our competitors mine a block, we can publish our two blocks in response. Since our chain is longer, their block (and work) will be wiped out by ours.

2. $1 - Q$ — Someone else finds the next block:

    1. $(1 - Q)Q$ — We find the next block after that. Again, we're two blocks ahead, and if we broadcast our block the competitions' block will be wiped out.
    2. $(1 - Q)^2$ — They find the next block. They're now two blocks ahead, so the simplest thing to do is switch to their chain, throwing away our block.

In the first two outcomes, we've made our competitors waste work on blocks that didn't get into the chain; in the last outcome our work was wasted. Thus, if our goal is to get more blocks than our competitors, we need the probability of the first two outcomes to be higher than the last outcome:

$$
\begin{aligned}
Q + (1 - Q)Q &> (1 - Q)^2 \\
2Q - Q^2 &> 1 - 2Q + Q^2 \\
-2Q^2 + 4Q - 1 &> 0
\end{aligned}
$$

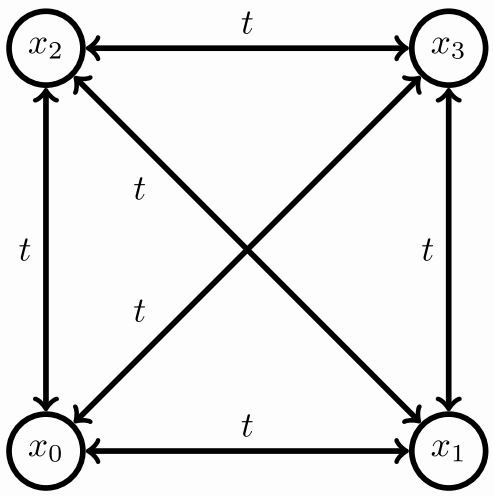
This is a quadratic, with roots $Q = 1 \pm \frac{1}{\sqrt{2}}$. One root is out-of-range ($Q > 1$), giving us:

$$
\begin{aligned}
Q &> 1 - \frac{1}{\sqrt{2}} \\
Q &> \ 29.2\%
\end{aligned}
$$

In short, if we have more than 29.2% hashing power the optimal strategy is to withhold, basically because when we find a block, our competitors have to find two blocks in a row to beat us

3.2 Non-Negligible Propagation Delays

What happens when block propagation time isn't negligible? The simplest case is the uniform, fully-connected, topology with all miners experiencing a uniform propagation delay t with respect to each other:



In this simplistic topology, when a miner finds a block they succeed in collecting their reward only if another miner didn't already find a block that they did not already know about; we win so long as no block was found in the time interval t. Mining is a Poisson process, so for a block interval T that probability is:

$$P\{N = 0\} = e^{-t/T} \approx \frac{t}{T}$$

For a propagation delay less than 25% of the block interval, the approximation $\frac{t}{T}$ introduces an error less than 4%.

Let's suppose we're a miner with non-negligible hashing power Q. We will *not* attempt any block withholding, and will immediately publish blocks when we find them. When we find a block, the following outcomes are possible:

1. $1 - \frac{t}{T}(1 - Q)$ — No other block is found.
2. $\frac{t}{T}(1 - Q)$ – Another miner found a block first. We don't however switch to that block, as if we find another block we can still get rewarded for both; switching guarantees that our block reward will be orphaned. The next block breaks the tie:
    1. $\frac{t}{T}(1 - Q)Q$ — We find another block before anyone else does, breaking the tie in our favour[8].
    2. $\frac{t}{T}(1 - Q)(1 - Q)$ — Someone else finds another block, breaking the tie against us.

We win in the first and second cases, and lose in the last, which lets us compute the (normalized) expected return of finding a block:

$$E_Q = 1 - \frac{t}{T}(1 - Q)^2$$

Let's take the derivative of the expected return with respect to hashing power:

$$\frac{dE_Q}{dQ} = 2\frac{t}{T}(1-Q)$$

Note how this derivative is positive for $0 < Q < 1$. This means that miners can always earn more money by centralizing: the fact that miners don't orphan their own block rewards gives them a head start. Put another way, all else being equal, larger miners earn more money per unit hashing power than smaller miners.

Similarly, we can show that larger miners' stale rates are affected less by propagation delays than smaller miners by looking at the derivative of the expected return with respect to propagation time:

$$\frac{dE_Q}{dt} = -\frac{1}{T}(1-Q)^2$$

For a given propagation time increase dt, a $Q = 25\%$ miner is about half as affected by propagation delays as a $Q = 0\%$ miner, and a $Q = 50\%$ miner only a quarter as affected.

3.2.1 Small Miners

What about the expected return of negligible hashing power miners? For them the existence of a large miner has a major change on their expected return: even if no-one else found a block just prior, they'll still lose if the large miner finds a block before the small miner's block gets to them, and then subsequently finds a second block.

This means if we're part of the $(1-Q)$ small miners, when we find a block the following outcomes are possible:

1. $\frac{t}{T}$ — Someone else found a block first. Since we have negligible hashing power, the chance of us finding a subsequent block in time is negligible, so we lose our block reward.

2. $1 - \frac{t}{T}$ — No-one else found a block first. We're now racing to get our block to the large miner:

    1. $(1 - \frac{t}{T})\frac{t}{T}Q$ — The large miner finds a block before they get ours. Small and large miners are tied:

        1. $(1 - \frac{t}{T})\frac{t}{T}QQ$ — Large miner finds a subsequent block, breaking the tie against us. We lose our reward.

        2. $(1 - \frac{t}{T})\frac{t}{T}Q(1-Q)$ — Small miners find a subsequent block, breaking the tie

in our favor; our block wins.

2. $(1 - \frac{t}{T})(1 - \frac{t}{T}Q)$ — Large miner doesn't find another block. A small miner may also find a block, but that's irrelevant to us because we have the head-start[9] we win.

Thus our expected return is:

$$E_R = \left(1 - \frac{t}{T}\right)\frac{t}{T}Q(1 - Q) + \left(1 - \frac{t}{T}\right)\left(1 - \frac{t}{T}Q\right)$$
$$E_R = \left(1 - \frac{t}{T}\right)\left(1 - \frac{t}{T}Q^2\right)$$

Let's take the derivative with respect to the hashing power of the large miner:

$$\frac{dE_R}{dQ} = 2Q\frac{t}{T^2}(t - T)$$

Since $T > t$, the derivative is always negative: as the larger miner's hashing power increases, the smaller miners revenue per block goes down. Of course, this isn't surprising: we already showed that larger miners make excess super-linear returns; those returns have to come from somewhere.

Similarly, we can show that longer propagation times t always negatively impact small miners expected returns:

$$\frac{dE_R}{dt} = \frac{Q^2(2t - T) - T}{T^2}$$

Again, since $T > t$, and $0 < Q < 1$, the derivative is always negative and longer propagation times decrease expected returns for small miners.

3.2.2 Expected Return Ratio

Mining is a zero sum game: since difficulty adjusts downward if hashing power drops, pushing your competitors out of business makes you more profitable in the long run. So let's look at the ratio between large and small miners of their respective expected returns, E_Q/E_R.

$$\frac{E_Q}{E_R} = \frac{1 - \frac{t}{T}(1 - Q)^2}{(1 - \frac{t}{T})(1 - \frac{t}{T}Q^2)}$$

We already know that increasing Q gives the advantage to the larger miner at the expense

of the smaller miners; what about increasing the propagation delay t? By applying the product rule we see that:

$$\frac{d\frac{E_Q}{E_R}}{dt} = \frac{\frac{dE_Q}{dt}E_R - \frac{dE_R}{dt}E_Q}{(E_Q)^2}$$

Since $\frac{dE_R}{dt} < 0$, and all other terms are positive, increasing the propagation delay t always increases the large miner's revenue relative to the small miners.

3.3 Non-Uniform Propagation Delays

The real Bitcoin network isn't uniform: some miners receive blocks faster than others, in part for the simple reason that not all miners - or nodes - have the same bandwidth. In addition, latency between miners isn't uniform due to speed-of-light delays between miners in different geographic locations (a very good thing for decentralization!).

How do these non-uniform delays affect expected returns? Do those non-uniform delays give some miners an advantage over others?

Suppose we're a miner and we find a block. We'll say that that set of miners mining our block at time $0 \leq t \leq \infty$ is Q_t, and the set of miners who are not mining our block is R_t, allowing us to define $\gamma(t)$, the fraction of hashing power in Q at a given moment:

$$\gamma(t) = \frac{\sum Q_t}{\sum R_t + \sum Q_t}$$

We want to know if there exist $\gamma(t)$ such that the stale rate for miners in R is greater than Q.

As with our block withholding case, Q miners need to find one more block before R miners find two blocks to avoid their block reward getting orphaned; the converse is true for R miners. To simplify the analysis we'll assume that the next block Q miners find propagates approximately instantaneously - an optimization that miners do in fact use, by mining empty blocks for the first few seconds after a block is found. Similarly, we'll assume the best-possible-case where R blocks propagate instantaneously. Since R blocks propagate instantaneously, once a R block is found $\gamma(t)$ becomes a constant function, as miners mine the most-work chain they see first.

Let T_{Q_i}, T_{R_i} be the arrival times of the i-th Q and R blocks, $N(t)$ the total number of blocks found at time t, and $\lambda = \frac{1}{T}$. We start at $T_{Q_0} = t = 0$. The expected return for R miners is the probability they will find two blocks before Q finds their second block:

$$
\begin{aligned}
E_R &= P\{T_{R_0} < T_{R_1} < T_{Q_1}\} \\
&= \int_0^\infty P\{T_{R_0} = t | N(t) = 0\}\, P\{N(t) = 0\}\, P\{T_{R_1} < T_{Q_1}\}\, dt \\
&= \int_0^\infty \lambda\Big(1 - \gamma(t)\Big) e^{-\lambda t}\Big(1 - \gamma(t)\Big)\, dt \\
&= \lambda \int_0^\infty e^{-\lambda t}\Big(1 - \gamma(t)\Big)^2\, dt
\end{aligned}
$$

The expected return for Q miners is the probability they'll either find one more block before R, or failing that, find a second block before R finds their second block:

$$
\begin{aligned}
E_Q =& P\{T_{Q_1} < T_{R_0} \cap T_{R_0} < T_{Q_1} < T_{R_1}\} \\
&= \int_0^\infty P\{T_{Q_1} = t | N(t) = 0\}\, P\{N(t) = 0\} \\
&\qquad + P\{T_{R_0} = t | N(t) = 0\}\, P\{N(t) = 0\}\, P\{T_{R_0} < T_{Q_1} < T_{R_1}\}\, dt \\
&= \int_0^\infty e^{-\lambda t}\Big(\lambda\gamma(t) + \lambda\Big(1 - \gamma(t)\Big)\gamma(t)\Big)\, dt \\
&= \lambda \int_0^\infty e^{-\lambda t}\Big(2\gamma(t) - \gamma(t)^2\Big)\, dt
\end{aligned}
$$

Let's sanity check our work. We defined E_Q and E_R as the expected returns for the next block - the chance a block isn't stale. Since R miners' blocks propagate approximately instantly, any stales for Q should decrease the stale rate for R, resulting in an overall expected return per block of one:

$$
\begin{aligned}
E_Q + E_R =& \lambda \int_0^\infty e^{-\lambda t}\Big(2\gamma(t) - \gamma(t)^2\Big)\, dt + \lambda \int_0^\infty e^{-\lambda t}\Big(1 - \gamma(t)\Big)^2\, dt \\
=& 2\lambda \int_0^\infty e^{-\lambda t}\gamma(t)\, dt - \lambda \int_0^\infty e^{-\lambda t}\gamma(t)^2\, dt \\
&+ \lambda \int_0^\infty e^{-\lambda t}\gamma(t)^2\, dt - 2\lambda \int_0^\infty e^{-\lambda t}\gamma(t)\, dt + \lambda \int_0^\infty e^{-\lambda t}\, dt \\
=& \lambda \int_0^\infty e^{-\lambda t}\, dt \\
=& 1
\end{aligned}
$$

Secondly, if $\gamma(t)$ is constant, we're effectively saying that block propagation is instantaneous, and someone is doing a block withholding attack:

$$\gamma(t) = k \implies \begin{cases} E_R = \lambda \int_0^\infty e^{-\lambda t}(1-k)^2 \, dt \\ \quad = \lambda(1-k)^2 \int_0^\infty e^{-\lambda t} \, dt \\ \quad = (1-k)^2 \\ \\ E_Q = \lambda \int_0^\infty e^{-\lambda t}(2k-k^2) \, dt \\ \quad = \lambda(2k-k^2) \int_0^\infty e^{-\lambda t} \, dt \\ \quad = 2k-k^2 \end{cases}$$

Sure enough, we get the exact same equations that we derived earlier for the block withholding attack. Good!

3.3.1 Slow Blocks

Can miners influence the $\gamma(t)$ of the blocks they create? Absolutely! In addition to simply withholding blocks, and DoS attacking relay nodes, miners can create blocks that propagate slower than normal ("slow blocks") in two main ways:

1. Validation Time — Among other things it's currently possible to exploit flaws in the existing signature hashing algorithm by creating especially large transactions with many inputs; this flaw been triggered both by deliberate attacks, as well as by accident (F2Pool produced a number of UTXO "cleanup" blocks that took dozens of seconds to validate[10]). However miners exploiting these flaws are very obvious, and will likely be fixed sooner or later with soft-forks (segwit's signature hashing scheme fixes this[11], at least for segwit transactions).

2. (Non-Pre-Forwarded) Size — Moving bytes takes time. Schemes like compact blocks and FIBRE try to mitigate this by moving those bytes in advance - mainly by taking advantage of pre-broadcast transactions - but those schemes are trivially defeated by simply putting non-pre-forwarded data in blocks; our defence is the upper limit imposed by the block size limit.

It's important to remember that both techniques can be easily done with miners with less hashing power than the 29.2% threshold required for a block withholding attack, so long as block propagation is non-uniform; slow blocks are not isomorphic to block withholding attacks as they work with any amount of hashing power.

It's very easy for pre-forwarding defeats to happen by accident. For example, every time the standard transaction rules are expanded by some nodes/miners, miners who have not adopted the newly allowed transaction forms are potentially put at a disadvantage. Similarly, transaction malleability can defeat pre-forwarding, and malleability can be triggered both by attackers (who don't necessarily need any hashing power at all) as well as by mistake.

Even deliberate actions that defeat pre-forwarding aren't always clearly malicious "attacks"; pre-forwarding defeats often have plausible deniability. For example, BTCC[12] offers BlockPriority, a service that expedites confirmations for BTCC customers: BTCC's pool will mine even zero-fee transactions if they either pay a BTCC customer, or were generated by a BTCC customer. Zero-fee transactions aren't reliably - if at all - relayed on the P2P network, so blocks containing such transactions are containing non-pre-forwarded data, slowing down propagation of BTCC's blocks. So long as propagation is non-uniform - and it is certainly is due to validationless mining, as well as the Great Firewall of China - $\gamma(t)$ for BTCC's blocks will rise rapidly enough that the slower propagation times for the remaining miners give BTCC a long-term advantage relative to their competitors.

Is BTCC attacking Bitcoin? Personally, rather than answer that question with a "yes" or "no", I'd rather answer with a Bitcoin protocol design where it doesn't matter what they do to maximize their revenue; Bitcoin mining shouldn't have negative externalities.

4 Footnotes

1. By "miner" we mean someone with hashing power who also has a full node that creates blocks; by "hasher" we mean someone with hashing power who does no block validation themselves, instead pointing their hashing power at a pool. ↩

2. The stale rate is the percentage of blocks found that do not end up in the main chain due to propagation delays - other miners finding blocks first. This is commonly referred to by the misnomer "orphan rate", a term derived from the fact that in a stale block, the coinbase *transaction* has no valid "parent" block: it's now a worthless orphan whose outputs can't be spent. ↩

3. Note how in non-Bitcoin protocols with faster difficulty adjustment periods - such as Ethereum - the transition from subsidy revenue loss to revenue neutral happens much faster. ↩

4. "Mining cartel attack", Dec 12th 2010 ↩

5. …or possibly partly plagiarized. ↩

6. “Majority Is Not Enough”, Ittay Eyal and Emin Gün Sirer, Nov 15th 2013 ↩

7. “Re: [Bitcoin-development] we can all relax now”, Peter Todd, Nov 7th 2013, bitcoin-development mailing list ↩

8. This analysis is slightly simplified, as we’re not taking propagation into account for the tie-breaking scenario. However as the probability of multiple ties in a row very quickly diminishes the simplification is insignificant. ↩

9. Remember that a small miner has a negligible chance of finding the two blocks in a row required to beat us. ↩

10. “The Megatransaction: Why Does It Take 25 Seconds?”, Rusty Russell, July 8th 2015 ↩

11. “BIP143: Transaction Signature Verification for Version 0 Witness Program”, Johnson Lau and Pieter Wuille, Bitcoin Improvement Proposals, Accessed July 29th 2016 ↩

12. Disclosure: as of writing, BTCC sponsors my work on Bitcoin Core, along with Bitfinex, somewhat obligating me to use them as an example! ↩

Peter Todd

petertodd
petertoddbtc