Lecture 8: Scaling Throughput in Bitcoin

Principles of Blockchains, University of Illinois,
Professor: Pramod Viswanath
Scribe: Xuechao Wang

February 18, 2021

Abstract

In this lecture, we see why security inherently limits the mining rate and hence the throughput (transactions per second) of Bitcoin. Scaling the throughput while maintaining the high security of Bitcoin has been a major research topic of the past few years. In this lecture, we look at three major efforts with increasing success: GHOST (changes the fork choice rule in mining), Bitcoin-NG (a single successful miner can propose multiple blocks in sequence), and Prism (separates proposer blocks from transaction blocks, and yet mines them together via cryptographic sortition). GHOST is vulnerable to a balance attack that reduces its throughput back to Bitcoin levels. Bitcoin-NG is vulnerable to bribing attacks by an *adaptive* adversary, again restricting its throughput back to Bitcoin levels. Prism, closely related to Fruitchains from the previous lecture, retains the high security of Bitcoin and achieves throughput restricted only by the underlying network capacity.

Bitcoin's throughput is security limited

The only key parameters in Bitcoin are: mining difficulty (decides the mining rate λ) and block size (B). In practice, λ is set to be 0.1/minute (i.e., one block every ten minutes is mined on average) and B is roughly 1 MB (approximately 2000 transactions). So the throughput of Bitcoin (assuming only honest miners) is very low:

$$(1-\beta)\lambda B \approx 4 \text{ transactions/second},$$

representing a data rate of roughly 16 Kbps. We observe that the underlying network speeds are far larger: e.g., 100 Mbps is an inexpensive cable modem speed: so clearly, the network is not limiting Bitcoin's throughput. We can try increasing the two parameters λ, B to directly increase throughput:

- Increasing λ directly impacts the security: the hash power of the adversary that can be tolerated decreases as $\lambda\Delta$ increases (see Figure 1, reproduced here from Lecture 6). While λ can be increased somewhat, e.g., say by a factor of 40 (as in Ethereum, a blockchain that also uses the longest chain protocol) without impacting safety seriously, there is still an inherent limitation on the mining rate due to the impact on security.
- Increasing B proportionally increases the network delay Δ: after all, it takes longer to transmit a bigger packet. This basic relationship is captured by a common networking model:

  $$\Delta = \frac{B}{C} + D,$$

  where C is the network *capacity* (measured in bits/s) and D is the propagation delay (essentially the ratio of distance to the speed of light). Empirical measurements on the Bitcoin main

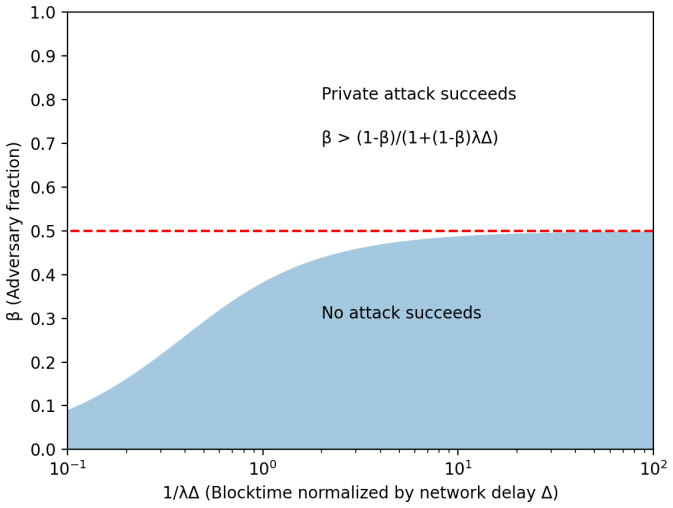

Figure 1: Minimum hash power needed by the adversary to succeed in any safety attack.

network has borne out this observation (see Figure 2). Thus increasing B has the same effect as increasing the mining rate λ: this increases the product $\lambda\Delta$, undermining the security.

We conclude that the throughput of Bitcoin is *security limited*. In this lecture, we see three separate attempts to increase the throughput of Bitcoin while maintaining its security; the first two are successful only to limited degrees while the third resolves the issue entirely.

GHOST fork choice rule

Increasing the mining rate and/or the block size has the direct effect of increasing the product $\lambda\Delta$ which is directly proportional to the *forking rate*: the fraction of blocks not on the longest chain. The forking rate, in turn, directly impacts the security of the longest chain protocol. Perhaps one should deviate from the longest chain protocol to a new mining rule that *embraces* forking (the longest chain protocol fights forking) – this is the spirit of the GHOST (greedy heaviest observed subtree) fork choice rule: the key idea is to measure the "quality" of a chain not by its height but by how heavy the weight of the subtree that supports the chain is. The GHOST rule is best explained by an example.

Consider the block tree in Figure 3, with substantial forking caused by a large mining rate. The adversary faces no forking (since its mining efforts are all synchronized) and thus finds it easier to overtake the longest chain via a private attack. The GHOST rule also dictates honest nodes to mine at the tip of a chain, but the chosen chain is potentially different from the longest chain as follows: for each block we calculate the *weight* of the subtree rooted at that block, i.e., the number of blocks in the subtree (including the block itself). Note that the weight of every leaf block is simply unity; the weights of the blocks are labeled inside each block in Figure 3. The GHOST chain is constructed by starting at the genesis and following the heaviest subtree whenever a fork is encountered; the GHOST chain (red/green) and the longest chain (blue) could be different, as in Figure 3.

Private attack on GHOST. Consider the adversary launching a private attack, i.e., mining in private and not participating in the public mining process of the honest nodes. The weight of the honest subtree (rooted at the genesis) is proportional to the honest mining rate, $(1-\beta)\lambda$, irrespective

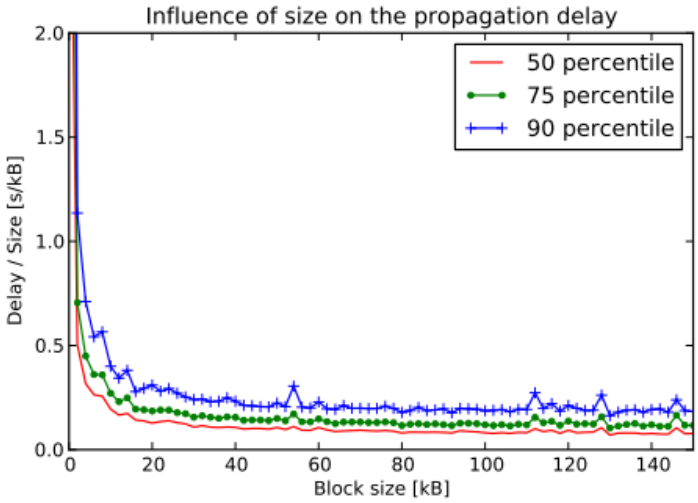

Figure 2: Plot of network delay as a function of the size of the block being communicated based on empirical measurements on the Bitcoin network.

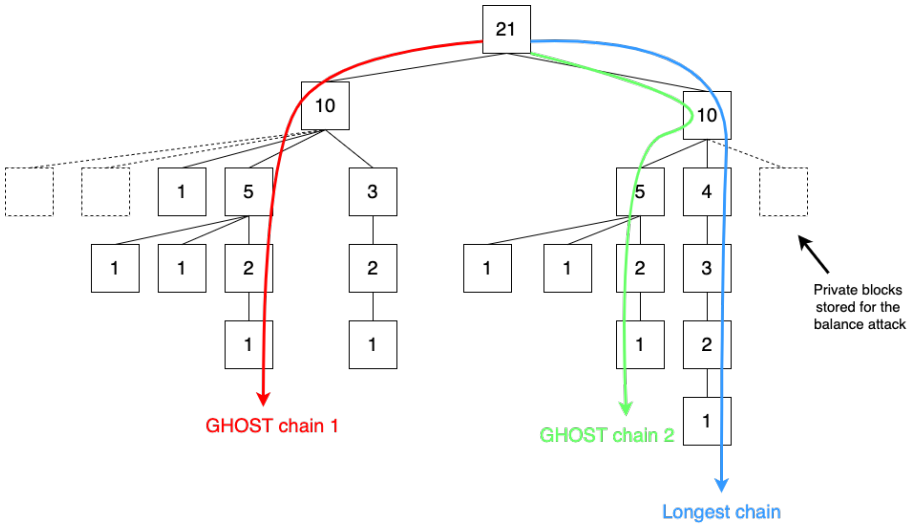

Figure 3: GHOST rule dictates mining to be conducted at the tip of the red/green colored chain, different from the longest chain rule (blue colored).

of the forking rate of the honest subtree. The weight of the private adversary chain (or any subtree) is proportional to $\beta\lambda$. Thus the safety of the GHOST rule against a private attack is ensured as long as $1-\beta > \frac{1}{2}$, i.e., a majority of the hash power is honest (follows protocol) – note that this is true regardless of the mining rate λ, so the throughput can be increased by increasing the mining rate, only limited by the network capacity. This security analysis is focused on a specific adversarial strategy: the private attack. In the security analysis of the longest chain protocol (cf. Lecture 6), we have seen that the private attack is exactly optimal for any confirmation depth k (when $\Delta = 0$) and optimal for large confirmation depths (when $\Delta > 0$). So it might seem that private attack analysis suffices in general for security of blockchain protocols; this was the premise on which GHOST was invented and its earlier security analysis was based. Unfortunately, this premise turns out to be false as we see next.

Balance attack on GHOST. The goal of the adversary in the balance attack is to maintain *two* growing chains that have equal weights (according to the GHOST rule). When there are two or more chains with equal weights, honest miners pick the earliest chain to mine on. By controlling the network enough so that half the honest nodes see one of the chains earlier than the other (and vice versa with the other half of honest nodes), the adversary ensures that half the honest nodes mine at the tip of one of the chains (red) and the other half mine at the tip of the other chain (green), *splitting* the honest mining power. The adversarial miner simply privately mines on the two blocks forking from the genesis itself (see Figure 3), making its blocks public as needed (to balance the two subtree weights). If the balance attack is kept up for longer than the confirmation depth k, then the safety of the blockchain protocol is fatally threatened. The following analysis sheds light on how much hash power β the adversary needs to possess to successfully maintain the balance attack.

Security analysis of balance attack on GHOST. Suppose at some steady state during the balance attack, within the network delay Δ, the left (right) subtree forked from genesis has grown N_ℓ (N_r) *honest* blocks, respectively. Due to the mining process and the split in honest power among the two subtrees, both N_ℓ and N_r are i.i.d. Poisson random variables with mean equal to $\frac{(1-\beta)\lambda\Delta}{2}$. For the balance attack to be successful, the adversary needs to ensure that N_ℓ and N_r match with high probability, i.e., it needs to mine $|N_\ell - N_r|$ blocks with high probability. Let N_a be the number of adversarial blocks mined in private (totally on both the left and right subtrees) within the network delay Δ, which is a Poisson random variable with mean $\beta\lambda\Delta$. We note that $\mathbb{E}[N_\ell - N_r]$ is zero, and the second moment of $N_\ell - N_r$ is

$$\mathbb{E}[(N_\ell - N_r)^2] = \mathrm{Var}(N_\ell - N_r) = \mathrm{Var}(N_\ell) + \mathrm{Var}(N_r) = (1-\beta)\lambda\Delta.$$

Then by Cauchy-Schwarz inequality, we have $\mathbb{E}(|N_\ell - N_r|) \le \sqrt{\mathbb{E}[(N_\ell - N_r)^2]} = \sqrt{(1-\beta)\lambda\Delta}$. For fixed $\beta > 0$, if $\lambda\Delta \gg 1$, then we have $\beta\lambda\Delta \gg \sqrt{(1-\beta)\lambda\Delta} \ge \mathbb{E}(|N_\ell - N_r|)$, i.e., the expected number of blocks the adversary can mine within the network delay Δ far exceeds the gap between the two subtrees. Hence, the adversary can easily balance the two subtrees and remain in the steady state of the balance attack. Once the balance attack is kept up for longer than the confirmation depth k, the safety of the GHOST rule is broken. Just like in Bitcoin, GHOST is not secure when $\lambda\Delta \gg 1$ for any β. Therefore, we see that the throughput of GHOST is again limited by security, due to the balance attack.

Bitcoin-NG

GHOST tried to speed up the mining process while using a different fork choice rule, but ran into the balance attack. Bitcoin-NG approaches the scaling problem from a different angle: separate the transactions (which make up all the throughput) from the headers (which provide the chaining and resultant security). The idea is similar in spirit to the Fruitchains protocol we saw in Lecture 7. There are two types of blocks: *proposer* blocks and *transaction* blocks. The mining process is described below; see Figure 4.

- Proposer blocks are mined as in Bitcoin, with the difference that the *public key of the miner* is included in the block; this is a departure from Bitcoin where the miners stay truly anonymous, which opens up security vulnerabilities as we will see shortly.
- The transaction blocks are not subject to proof of work and are simply signed (using the private key) by the miner whose public key is embedded in the immediately preceding proposer block. Since the transaction blocks are not inhibited by proof of work, they can be added to the blockchain as quickly as the underlying network can support their reliable broadcast. The transaction blocks continue to be mined until a new proposer block appears on the longest chain, after which only this new proposer can mine transaction blocks.
- Proposer blocks and transaction blocks are interspersed with each other: proposer blocks are mined at the tip of the longest chain: the length is only defined via the number of proposer blocks in the chain.

Security of Bitcoin-NG. The longest chain mining rule is executed only with respect to the proposer blocks. Further, confirmation of a proposer block (and all transaction blocks chain in between the proposer blocks on the chain up to the genesis block) occurs only when there are at least k *proposer* blocks mined underneath. Therefore, transaction blocks are not involved in the mining and confirmation processes; thus the security of the longest chain is provided by the proposer blocks and since the proposer blocks are mined infrequently, the same strong security guarantees as Bitcoin hold here.

Consider an *adaptive* adversary, i.e., one which chooses which participants to corrupt while the protocol is running. An adaptive adversary is typically implemented by coordinating with the participants outside the blockchain (e.g., on a website where participants can upload their blockchain status). Bitcoin has a strong *unpredictability* property which renders an adaptive adversary no more potent than a static one. The unpredictability property has two components: (a) which miner will be successful in the PoW process is unknown to anyone, including the miner, until the success is seen; (b) once a miner succeeds in the PoW process, the mined block is "sealed", i.e., the contents cannot be changed without rendering the nonce useless and, hence, successful mining defeated.

However, the presence of the public key of the miner in the proposer block identifies the miner of the following transaction blocks (until a new proposer block is mined), so property (a) above is violated. Further, the transaction blocks are no longer "sealed": their contents can be changed later by the owner of the private key associated with the immediately preceding proposer block; so property (b) above is also violated. An adaptive adversary can take advantage of the resulting predictability to launch the following *slow-down* attack. Whenever a proposer block is published, the adversary corrupts the miner and shuts down all following transaction blocks. Security is not threatened in this attack, but only the transactions inside of the proposer block get to the confirmed ledger and the resulting throughput is the same as that of Bitcoin.

Bitcoin-NG is an interesting design to scale throughput by separating payload (transactions) from security (longest chain rule of PoW mining and k-deep confirmation of proposer blocks). However, the order in which the two types of blocks, transaction and proposer, are linked leads to slow-down attacks by an adaptive adversary. A subtle change involving *reversing* the order of linking between transaction and proposer blocks directly solves this problem. This architecture is the same as the Fruitchains protocol from Lecture 7, but was independently constructed and its throughput optimality proved in Prism 1.0; this architecture completely solves the problem of throughput scaling in PoW longest chain protocols and is described next.

Prism: scaling throughput to physical limits

Consider the Fruitchains architecture from Lecture 7 (see Figure 5). By setting the transaction target difficulty easy, the transactions are published on the blockchain at a high throughput (limited only

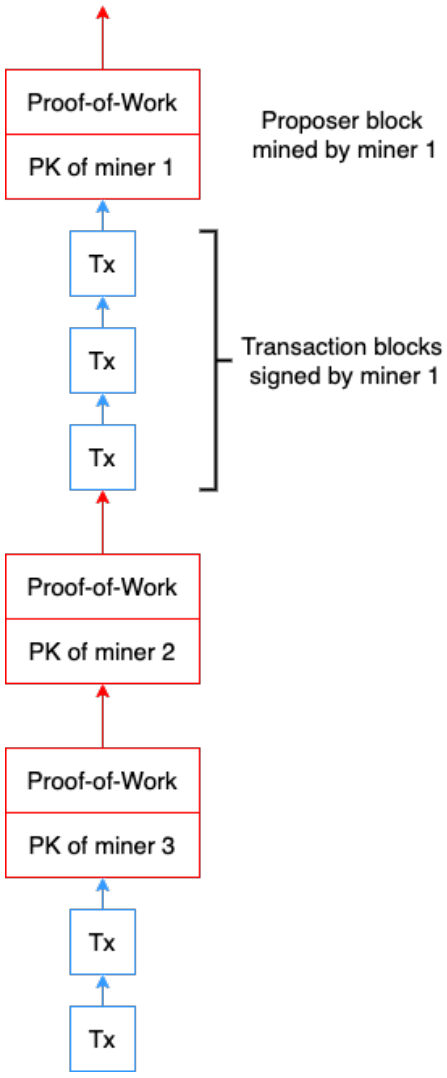

Figure 4: Bitcoin-NG protocol has two block types: proposer and transaction blocks, interspersed with each other. The proposer block is mined at the tip of the longest chain, as measured in terms of proposer blocks from the genesis. Transaction blocks are not subject to proof of work, but can only be signed using a private key associated with the public key as that identified in the immediately preceding proposer block.

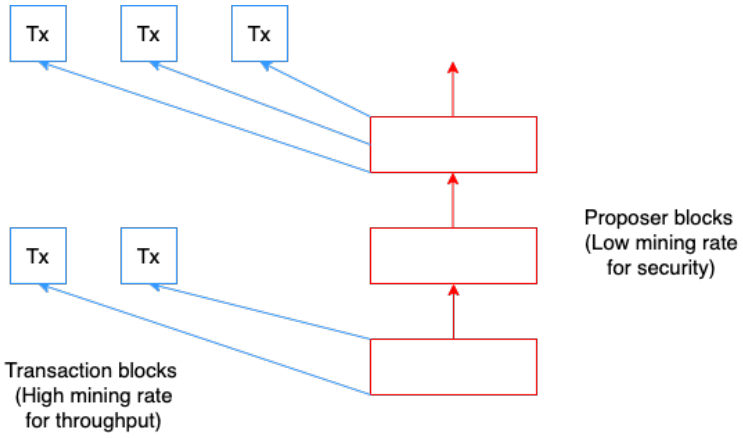

Figure 5: Prism 1.0 and Fruitchains protocols have two block types: proposer and transaction blocks, interspersed with each other. The transaction and proposer blocks are mined in parallel, coupled through the sortition process. Transaction block hashes are referred to by proposer blocks thus bringing the transactions into the confirmed ledger. Transaction block mining is kept easy (so throughput is high) and proposer block mining is kept difficult (so security is high).

by the underlying network capacity). By setting the proposer target difficulty hard, the security is maintained. This optimal throughput property of Fruitchains architecture was observed in an independent work that constructed a PoW protocol called Prism. It is interesting to compare and contrast with the Bitcoin-NG architecture (see Figure 4). The coupling between the transaction and proposer blocks is *reversed*: in Bitcoin-NG, a proposer block is mined first and then transaction blocks spawned from it. In Fruitchains, the previously mined transaction blocks are referred to by a proposer block (by inserting their hashes inside the proposer block). Further, the mining process of the two types of blocks are coupled in different ways: in Bitcoin-NG, the proposer block publishes its public key inside the block, whose corresponding private key is then used to sign the following transaction blocks – at the expense of allowing a slow-down attack by an adaptive adversary. In Fruitchains, the transaction and proposer blocks are mined *together* via the "two for one" mining procedure and cryptographic sortition (discussed in detail in Lecture 7) – this format of mining allows the overall architecture to have the same unpredictability properties as Bitcoin: until a superblock (containing both proposer and transaction blocks) is successfully mined, it is unclear whether the superblock will be identified as a proposer block or transaction block (property (a) above). Further, once the superblock is mined, the contents are sealed by the proof of work (property (b) above). Thus the security of Fruitchains is as high as that in Bitcoin.

Due to the decoupling of transaction blocks from the security of the protocol, the throughput is only restricted by the capacity of the underlying P2P network. A full-stack UTXO implementation of Prism achieves more than 70,000 transactions/second, as reported in this paper. The full Prism protocol also resolves another important limitation of the longest chain protocol: latency of confirmation. This is the topic of the next lecture.

References

Scaling throughput of blockchain protocols has been a major research focus in the past three years, arguably the most important issue studied in blockchains. One major direction has been to improve the throughput of PoW blockchains; this lecture has covered the highlights of this research thrust, ending with the throughput optimality of Fruitchains and Prism. One major direction we have not

covered in this lecture involves embracing forking by embedding appropriate reference links among the blocks (other than the usual parent reference link) to create a directed acyclic graph (DAG) structured blockchain. Each block in a DAG can reference multiple previous blocks instead of a unique ancestor (as in Bitcoin). The pertinent challenges are how to choose the reference links and how to obtain a total ordering of blocks from the observed DAG in a way that is secure. In a family of protocols, Inclusive, Spectre, and Phantom, every block references all previous orphan blocks. These reference links are interpreted in differing ways to give these different protocols.

For example, in Inclusive, the key observation is that the reference link structure provides enough information to emulate any main-chain protocol, such as the longest-chain or GHOST protocol, while in addition providing the ability to pull in stale blocks into a valid ledger. However, the security guarantee remains the same as that of Bitcoin (namely, tending to zero as the mining rate grows), and the throughput is security-limited. Conflux is another DAG-based protocol whose goal is to increase throughput. However, Conflux's reference links are not used to determine where to mine blocks or how to confirm them; they are only used to include side-chain blocks into the main chain to improve throughput. The main chain itself is selected by the GHOST rule. Due to the vulnerability of GHOST to the balance attack, the secured throughput of Conflux is limited to Bitcoin levels. Although Prism uses a DAG to order transactions, it diverges from prior DAG schemes by separating block proposals from block ordering in the protocol. This helps because an adversarial party that misbehaves during block proposal does not affect the security of transaction ordering, and vice versa; it provides a degree of algorithmic sandboxing.