Bitcoin-NG: A Scalable Blockchain Protocol

Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse, ***Cornell University***

https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal

This paper is included in the Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16).

March 16–18, 2016 • Santa Clara, CA, USA

ISBN 978-1-931971-29-4

Open access to the Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16) is sponsored by USENIX.

Bitcoin-NG: A Scalable Blockchain Protocol*

Ittay Eyal Adem Efe Gencer Emin Gün Sirer Robbert van Renesse
Cornell University

Abstract

Cryptocurrencies, based on and led by Bitcoin, have shown promise as infrastructure for pseudonymous online payments, cheap remittance, trustless digital asset exchange, and smart contracts. However, Bitcoin-derived blockchain protocols have inherent scalability limits that trade off between throughput and latency, which withhold the realization of this potential.

This paper presents Bitcoin-NG (Next Generation), a new blockchain protocol designed to scale. Bitcoin-NG is a Byzantine fault tolerant blockchain protocol that is robust to extreme churn and shares the same trust model as Bitcoin.

In addition to Bitcoin-NG, we introduce several novel metrics of interest in quantifying the security and efficiency of Bitcoin-like blockchain protocols. We implement Bitcoin-NG and perform large-scale experiments at 15% the size of the operational Bitcoin system, using unchanged clients of both protocols. These experiments demonstrate that Bitcoin-NG scales optimally, with bandwidth limited only by the capacity of the individual nodes and latency limited only by the propagation time of the network.

1 Introduction

Bitcoin has emerged as the first widely-deployed, decentralized global currency, and sparked hundreds of copycat currencies. Overall, cryptocurrencies have garnered much attention from the financial and tech sectors, as well as academics; achieved wide market penetration in underground economies [38]; reached a $12B market cap; and attracted close to $1B in venture capital [15]. The core technological innovation powering these systems is the *Nakamoto consensus* protocol for maintaining a distributed ledger known as the blockchain. The blockchain technology provides a decentralized, open, Byzantine fault-tolerant transaction mechanism, and promises to become the infrastructure for a new generation of Internet interaction, including anonymous online payments [14], remittance, and transaction of digital assets [16]. Ongoing work explores smart digital contracts, enabling anonymous parties to programmatically enforce complex agreements [31, 56].

Despite its potential, blockchain protocols face a significant scalability barrier [51, 36, 19, 5]. The maximum rate at which these systems can process transactions is capped by the choice of two parameters: block size and block interval. Increasing block size improves throughput, but the resulting bigger blocks take longer to propagate in the network. Reducing the block interval reduces latency, but leads to instability where the system is in disagreement and the blockchain is subject to reorganization. Bitcoin currently targets a conservative 10 minutes between blocks, yielding 10-minute expected latencies for transactions to be encoded in the blockchain. The block size is currently set at 1MB, yielding only 1 to 3.5 transactions per second for Bitcoin for typical transaction sizes. Proposals for increasing the block size are the topic of heated debate within the Bitcoin community [47].

In this paper, we present Bitcoin-NG, a scalable blockchain protocol, based on the same trust model as Bitcoin. Bitcoin-NG's latency is limited only by the propagation delay of the network, and its bandwidth is limited only by the processing capacity of the individual nodes. Bitcoin-NG achieves this performance improvement by decoupling Bitcoin's blockchain operation into two planes: *leader election* and *transaction serialization*. It divides time into epochs, where each epoch has a single leader. As in Bitcoin, leader election is performed

*The authors are supported in part by AFOSR grants FA2386-12-1-3008, F9550-06-0019, by the AFOSR MURI Science of Cyber Security: Modeling, Composition, and Measurement as AFOSR grant FA9550-11-1-0137, by NSF grants CNS-1601879, 0430161, 0964409, 1040689, 1047540, 1518779, 1561209, and CCF-0424422 (TRUST), by ONR grants N00014-01-1-0968 and N00014-09-1-0652, by DARPA grants FA8750-10-2-0238 and FA8750-11-2-0256, by MDCN/iAd grant 54083, and by grants from Microsoft Corporation, Facebook Inc., and Amazon.com.

randomly and infrequently. Once a leader is chosen, it is entitled to serialize transactions unilaterally until a new leader is chosen, marking the end of the former's epoch.

While this approach is a significant departure from Bitcoin's operation, Bitcoin-NG maintains Bitcoin's security properties. Implicitly, leader election is already taking place in Bitcoin. But in Bitcoin, the leader is in charge of serializing history, making the entire duration of time between leader elections a long system freeze. In contrast, leader election in Bitcoin-NG is forward-looking, and ensures that the system is able to continually process transactions.

Evaluating the performance and functionality of new consensus protocols is a challenging task. To help perform this quantitatively and provide a foundation for the comparison of alternative consensus protocols, we introduce several metrics to evaluate implementations of Nakamoto consensus. These metrics capture performance metrics such as protocol goodput and latency, as well as various aspects of its security, including its ability to maintain consensus and resist centralization.

We evaluate the performance of Bitcoin-NG on a large emulation testbed consisting of 1000 nodes, amounting to over 15% of the current operational Bitcoin network [41]. This testbed enables us to run unchanged clients, using realistic Internet latencies. We compare Bitcoin-NG with the original Bitcoin client, and demonstrate the critical trade-offs inherent in the original Bitcoin protocol. Controlling for network bandwidth, reducing Bitcoin's latency by decreasing the block interval and improving its throughput by increasing the block size both yield adverse effects. In particular, fairness suffers, giving large miners an advantage over small miners. This anomaly leads to centralization, where the mining power tends to be concentrated under a single controller, breaking the basic premise of the decentralized cryptocurrency vision. Additionally, mining power is lost, making the system more vulnerable to attacks. In contrast, Bitcoin-NG improves latency and throughput to the maximum allowed by network conditions and node processing limits, while avoiding the fairness and mining power utilization problems.

In summary, this paper makes three contributions. First, it outlines the Bitcoin-NG scalable blockchain protocol, which achieves significantly higher throughput and lower latency than Bitcoin while maintaining the Bitcoin trust assumptions. Second, it introduces quantitative metrics for evaluating Nakamoto consensus protocols. These metrics are designed to ground the ongoing discussion over parameter selection in Bitcoin-derived currency. Finally, it quantifies, through large-scale experiments, Bitcoin-NG's robustness and scalability.

2 Model and Goal

The system is comprised of a set of nodes $\mathcal{N}$ connected by a reliable peer-to-peer network. Each node can poll a random oracle [6] as a random bit source. Nodes can generate key-pairs, but there is no trusted public key infrastructure.

The system employs a cryptopuzzle system, defined by a cryptographic hash function H. The solution to a puzzle defined by the string y is a string x such that $H(y|x)$ — the hash of the concatenation of the two — is smaller than some target. Each node i has a limited amount of compute power, called *mining power*, measured by the number of potential puzzle solutions it can try per second. A solution to a puzzle constitutes a *proof of work*, as it statistically indicates the amount of work a node had to perform in order to find it.

At any time t, a subset of nodes $B(t) \subset \mathcal{N}$ are Byzantine and behave arbitrarily, controlled by a single adversary. The other nodes are *honest* — they abide by the protocol. The mining power of each node i is $m(i)$. The mining power of the Byzantine nodes is less than 1/4 of the total compute power at any given time:

$$\forall t : \sum_{b \in B(t)} m(b) < \frac{1}{4} \sum_{n \in \mathcal{N}} m(n)$$

because proof-of-work blockchains, Bitcoin-NG included, are vulnerable to selfish mining by attackers larger than $1/4$ of the network [25].

Nakamoto Consensus

The nodes are to implement a replicated state machine (RSM) [33, 50]. Properties of the system can be compared to those of classical consensus [46]:

Termination There exists a time difference function $\Delta(\cdot)$ such that, given a time t and a value $0 < \varepsilon < 1$, the probability is smaller than ε that at times $t', t'' > t + \Delta(\varepsilon)$ a node returns two different states for the machine at time t.

Agreement There exists a time difference function $\Delta(\cdot)$ such that, given a $0 < \varepsilon < 1$, the probability that at time t two nodes return different states for $t - \Delta(\varepsilon)$ is smaller than ε.

Validity If the fraction of mining power of Byzantine nodes is bounded by f, i.e., $\forall t : \frac{\sum_{b \in B(t)} m(b)}{\sum_{n \in \mathcal{N}} m(n)} < f$, then the average fraction of state machine transitions that are not inputs of honest nodes is smaller than f.

3 Bitcoin and its Blockchain Protocol

Bitcoin is a distributed, decentralized crypto-currency [7, 8, 9, 43], which implicitly defined and implemented

Nakamoto consensus. Bitcoin uses the blockchain protocol to serialize transactions of the Bitcoin currency among its users. The replicated state machine maintains the balances of the different users, and its transitions are transactions that move funds among them. This state machine is managed by the system nodes, called miners.

Each user commands *addresses*, and sends Bitcoins by forming a transaction from her address to another's address and sending it to the nodes. More explicitly, a transaction is from the output of a previous transaction to a specific address. An output is *spent* if it is the input of another transaction. A client owns x Bitcoins at time t if the aggregate of unspent outputs to its address is x. Transactions are protected with cryptographic techniques that ensure only the rightful owner of a Bitcoin address can transfer funds from it. Miners accept transactions only if their sources have not been spent, thereby preventing users from double-spending their funds. The miners commit the transactions into a global append-only log called the *blockchain*.

The blockchain records transactions in units of blocks. Each block includes a unique ID, and the ID of the preceding block. The first block, dubbed *the genesis block*, is defined as part of the protocol. A valid block contains (1) a solution to a cryptopuzzle involving the hash of the previous block, (2) the hash (specifically, the Merkle root) of the transactions in the current block, which have to be valid, and (3) a special transaction, called the *coinbase*, crediting the miner with the reward for solving the cryptopuzzle. This process is called Bitcoin *mining*, and, by slight abuse of terminology, we refer to the creation of blocks as *block mining*. The specific cryptopuzzle is a double-hash of the block header whose result has to be smaller than a set value. The *problem difficulty*, set by this value, is dynamically adjusted such that blocks are generated at an average rate of one every ten minutes.

Mining When a miner creates a block, she is compensated for her efforts with Bitcoins. This compensation includes a per-transaction fee paid by the users whose transactions are included, as well as an amount of new Bitcoins that did not exist before.

Forks Any miner may add a valid block to the chain by simply publishing it over an overlay network to all other miners. If multiple miners create blocks with the same preceding block, the chain is *forked* into *branches*, forming a tree. Other miners may subsequently add new valid blocks to any of these branches. When a miner tries to add a new block after an existing block, we say it *mines on* the existing block. If this block is a leaf of a branch, we say he mines on the branch.

To resolve forks, the protocol prescribes on which chain the miners should mine. The criterion is that the winning chain is the *heaviest one*, that is, the one that required (in expectancy) the most mining power to generate. All miners add blocks to the heaviest chain of which they know, with random tie-breaking. We note that choosing a longest branch at random is suggested by Eyal and Sirer [25]. The operational client currently chooses the first branch it has heard of, making it more vulnerable in the general case. The heaviest chain a node knows is the serialization of RSM inputs it knows, and hence describes the RSM's state. The formation of forks is undesirable, as they indicate that there is no globally-agreed RSM state.

Branches and blocks outside the main chain are called pruned (and not orphans, as is common in informal discussions, since they have a parent in the block tree). Transactions in pruned blocks are ignored. They can be placed in the main chain at any later time, unless a contradicting transaction (that spends the same outputs) was placed there in the meantime.

Block dissemination over the Bitcoin overlay network takes seconds, whereas the average mining interval is ten minutes. Therefore, accidental bifurcation occurs on average about once every 60 blocks [18].

We are now ready to describe Bitcoin-NG.

4 Bitcoin-NG

Bitcoin-NG is a blockchain protocol that serializes transactions, much like Bitcoin, but allows for better latency and bandwidth without sacrificing other properties.

The protocol divides time into epochs. In each epoch, a single leader is in charge of serializing state machine transitions. To facilitate state propagation, leaders generate blocks. The protocol introduces two types of blocks: *key blocks* for leader election and *microblocks* that contain the ledger entries. Each block has a header that contains, among other fields, the unique reference of its predecessor; namely, a cryptographic hash of the predecessor header.

We detail the operation of the protocol in this section and explain its incentive system in Section 5.

4.1 Key Blocks and Leader Election

Key blocks are used to choose a leader. Like a Bitcoin block, a key block contains the reference to the previous block (either a key block or a microblock, usually the latter), the current Unix time, a coinbase transaction to pay out the reward, a target value, and a nonce field containing arbitrary bits. As in Bitcoin, for a key block to be valid, the cryptographic hash of its header must be smaller than the target value. Unlike Bitcoin, a key block contains a public key that will be used in subsequent microblocks.

As in Bitcoin, for a miner to generate a key block, it must iterate through nonce values until the crypto-puzzle condition is met. Consequently, the interval between

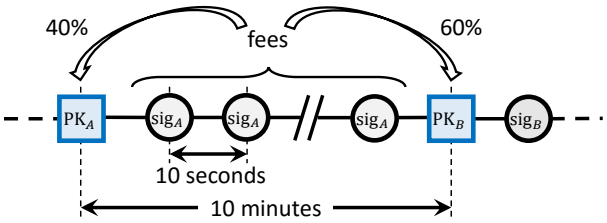

Figure 1: Structure of the Bitcoin-NG chain. Microblocks (circles) are signed with the private key matching the public key in the last key block (squares). Fee is distributed 40% to the leader and 60% to the next one.

consecutive key blocks is exponentially distributed. To maintain a set average rate, the difficulty is adjusted by deterministically changing the target value based on the Unix time in the key block headers.

In case of a fork, just as in Bitcoin, the nodes pick the branch with the most work, aggregated over all key blocks, with random tie breaking.

4.2 Microblocks

Once a node generates a key block it becomes the leader. As a leader, the node is allowed to generate microblocks at a set rate smaller than a predefined maximum. The maximum rate is deterministic, and can be much higher than the average interval between key blocks. The size of microblocks is bounded by a predefined maximum. Specifically, if the timestamp of a microblock is in the future, or if its difference with its predecessor's timestamp is smaller than the minimum, then the microblock is invalid. This bound prohibits a leader (malicious, greedy, or broken) from swamping the system with microblocks.

A microblock contains ledger entries and a header. The header contains the reference to the previous block, the current Unix time, a cryptographic hash of its ledger entries, and a cryptographic signature of the header. The signature uses the private key that matches the public key in the latest key block in the chain. For a microblock to be valid, all its entries must be valid according to the specification of the state machine, and the signature has to be valid. Figure 1 illustrates the structure.

Note that microblocks do not affect the weight of the chain, as they do not contain proof of work. This is critical for keeping the incentives aligned, as explained in Section 5.

4.3 Confirmation Time

When a miner generates a key block, he may not have heard of all microblocks generated by the previous leader. If microblock generation is frequent, this can be the common case on leader switching. The result is a short microblock fork, as illustrated in Figure 2. Such a fork is observed by any node that receives the

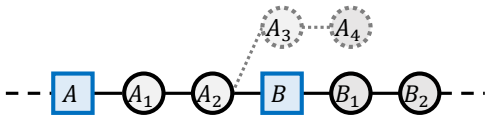

Figure 2: When microblocks are frequent, short forks occur on almost every leader switch.

to-be-pruned microblock (blocks A_3 and A_4 in the figure) before the new key block (block B in the figure). It is resolved once the key block propagates to that node. Therefore, a user that sees a microblock should wait for the propagation time of the network before considering it in the chain, to make sure it is not pruned by a new key block.

4.4 Remuneration

To motivate mining, a leader is compensated for her efforts by the protocol. Remuneration is comprised of two parts. First, each key block entitles its generator a set amount. Second, each ledger entry carries a fee. This fee is split by the leader that places this entry in a microblock, and the subsequent leader that generates the next key block. Specifically, the current leader earns 40% of the fee, and the subsequent leader earns 60% of the fee, as illustrated in Figure 1. The choice of this distribution is explained in Section 5.

In practice, the remuneration is implemented by having each key block contain a single coinbase transaction that mints new coins and deposits the funds to the current and previous leaders. As in Bitcoin, this transaction can only be spent after a maturity period of 100 key blocks, to avoid non-mergeable transactions following a fork.

4.5 Microblock Fork Prevention

Since microblocks do not require mining, they can be generated cheaply and quickly by the leader, allowing it to split the brain of the system, publishing different replicated-state-machine states to different machines. This allows for double spending attacks, where different nodes believe the same coins were spent with different transactions.

To demotivate such behavior, we use a dedicated ledger entry that invalidates the revenue of fraudulent leaders. Past work has used such entries in different contexts [22, 4, 13]. In Bitcoin-NG, the entry is called a *poison transaction*, and it contains the header of the first block in the pruned branch as a *proof of fraud*. The poison transaction has to be placed on the blockchain within the maturity window of the misbehaving leader's key block, and before the revenue is spent by the malicious leader. Besides invalidating the compensation sent to the leader that generated the fork, a poison transaction grants the current leader a fraction of that compensation,

e.g., 5%. The choice of this value is explained in Section 5.

Only one poison transaction can be placed per cheater, even if the cheater creates many forks. The cheater's revenue that is not relayed to the poisoner is lost.

5 Security Analysis

5.1 Incentives

This section describes how miners with capacity smaller than $1/4$ of the total network are incentivized to follow the protocol. Specifically, miners are motivated to (1) include transactions in their microblocks, (2) extend the heaviest chain, and (3) extend the longest chain. Unlike in Bitcoin, the latter two points are not identical.

Heaviest Chain Extension The motivation for extending the heaviest chain is the same as in Bitcoin. Since the honest majority will extend the heaviest chain, it will remain the main chain with high probability. A dishonest majority may arbitrarily switch to any branch and win [32]. A minority choosing to mine on another branch will not catch up with an honest majority, therefore it will mine on the main chain to ensure its revenues. We therefore argue that the guarantees of Bitcoin-NG are similar to those of Bitcoin [40] with respect to the Termination and Agreement properties of Nakamoto consensus.

Microblocks carry no weight, not even as a secondary index. If they did, it would increase the system's vulnerability to selfish mining [24, 44, 49]. In selfish mining, an attacker withholds blocks it has mined and publishes them judiciously to obtain a superior presence in the main chain. If microblocks carried weight, an attacker could keep secret microblocks and gain advantage by mining on microblocks unpublished to anyone else.

We conclude that Bitcoin-NG does not introduce a new vulnerability to selfish mining strategies, and so Bitcoin-NG is resilient to selfish mining against attackers with less than $1/4$ of the mining power. We therefore argue that the guarantees of Bitcoin-NG are similar to those of Bitcoin with respect to the validity property of Nakamoto consensus.

Transaction Inclusion A leader earns 40% of a transaction's revenue by placing it in a microblock. However, he could potentially improve his revenue by secretly trying to earn 100% of the fee. To do so, first, the leader creates a microblock with the transaction, but does not publish it. Then, he tries to mine on top of this secret microblock, while other miners mine on older microblocks. If the leader succeeds in mining the subsequent key block, he obtains 100% of the transaction fees. Otherwise, he waits until the transaction is placed in a microblock by another miner and tries to mine on top of it.

Consider a miner whose mining power ratio out of all mining power in the system is α. Denote by r_{leader} the revenue of the leader from a transaction, leaving $(1-r_{\text{leader}})$ for the next miner. In Bitcoin-NG, we have $r_{\text{leader}} = 40\%$. The value of r_{leader} has to be such that the average revenue of a miner trying the above attack is smaller than his revenue placing the transaction in a public microblock as it should:

$$\overbrace{\alpha \times 100\%}^{\text{Win 100\%}} + \overbrace{(1-\alpha) \times \alpha \times (100\% - r_{\text{leader}})}^{\text{Lose 100\%, but mine after txn}} < r_{\text{leader}} ,$$

therefore $r_{\text{leader}} > 1 - \frac{1-\alpha}{1+\alpha-\alpha^2}$. Assuming the power of an attacker is bounded by $1/4$ of the mining power, we obtain $r_{\text{leader}} > 37\%$, hence $r_{\text{leader}} = 40\%$ is within range.

Longest Chain Extension To increase his revenue from a transaction, a miner could avoid the transaction's microblock and mine on a previous block. Then he would place the transaction in its own microblock and try mining the subsequent key block. His revenue in this case must be smaller than his revenue by mining on the transaction's microblock as prescribed:

$$\overbrace{r_{\text{leader}}}^{\text{Place in microblock}} + \overbrace{\alpha(100\% - r_{\text{leader}})}^{\text{Mine next key block}} < \overbrace{100\% - r_{\text{leader}}}^{\text{Mine on existing microblock}} ,$$

therefore $r_{\text{leader}} < \frac{1-\alpha}{2-\alpha}$. Assuming the power of an attacker is bounded by $1/4$ of the mining power, we obtain $r_{\text{leader}} < 43\%$, hence $r_{\text{leader}} = 40\%$ is within range.

Optimal Network Assumption Incentive compatibility cannot be maintained in Bitcoin-NG for an attacker larger than about 29%. For larger attackers, the intersection of the two conditions is empty. But this limit does not come into play in the general case, where Bitcoin-NG, like Nakamoto's blockchain with random tie breaking [25], are secure only against attackers smaller than 23.2% [49] due to selfish mining attacks.

However, under optimal network assumptions, Bitcoin's blockchain is more resilient than Bitcoin-NG: Assuming a zero latency network where an attacker cannot rush messages — i.e., receive a message and send its own such that other nodes receive the attacker's message before the original one — Bitcoin is believed to be secure against selfish mining attackers of size up to almost $1/3$.

Bypassing Fee Distribution We note that a user can circumvent the $40-60\%$ transaction fee distribution by paying no transaction fee, and instead paying the current leader directly, using the coinbase address of the leader's key block. However, a user does not gain a significant advantage by doing so. As we have seen above, paying only the current leader increases the direct motivation of the current leader to place the transaction in a microblock, but reduces the motivation of future miners to mine on

this microblock. Moreover, if the leader does not include the transaction before the end of its epoch, subsequent leaders will have no motivation to place the transaction.

Other motives for fee manipulation, such as paying a large fee to encourage miners to choose a certain branch after a fork, apply to Bitcoin as well as Bitcoin-NG, and are outside the scope of this work.

5.2 Other concerns

Wallet Security The possibility of placing a poison transaction allows an attacker that obtains a leader's private key to revoke his revenue retroactively and earn a small amount. However, such an attacker is better off trying to steal the full leader's revenue when it becomes available, therefore the introduction of the poison transaction does not add a significant vulnerability.

Censorship Resistance A central goal of Bitcoin is to prevent a malicious discriminating miner from dropping a user's transactions. Censorship resistance is not impacted by the frequent microblocks of Bitcoin-NG.

First, we note that a leader's absolute power is limited to his epoch of leadership. A malicious leader can perform a DoS attack by placing no transactions in microblocks. Similarly, a benign leader that crashes during his epoch of leadership will publish no microblocks. Their influence ends once the next leader publishes his key block. The impact of such behaviors is therefore similar to that in Bitcoin, where nodes may mine empty blocks, but rarely do.

Assuming an honest majority and no backlog, a user will have her transaction placed in the first block generated by an honest miner. Since at least $3/4$ of the blocks are generated by honest miners, the user will have to wait for $4/3$ blocks on average, or 13.33 minutes. Key block intervals can be set to a rate that would reduce censorship to the minimum allowed by the network without incurring prohibitive deterioration of other metrics.

Resilience to Mining Power Variation Following Bitcoin's success, hundreds of alternative currencies were created [57], most with Bitcoin's exact blockchain structure, and many with the same proof-of-work mechanism. To maintain a stable rate of blocks, different instances of the blockchain tune their proof of work difficulty at different rates: Bitcoin once every 2016 blocks – about 2 weeks, Litecoin [37] every 2016 blocks (produced at a higher rate) – about 3.5 days, and Ethereum [56] on every block – about 12 seconds. However, whichever adjustment rate is chosen, these protocols are all sensitive to sudden mining power drops. Such drops happen when miners are incentivized to stop mining due to a drop in the currency's exchange rate, or to mine for a different currency that becomes more profitable due to a change in mining difficulty or exchange rate of either currency.

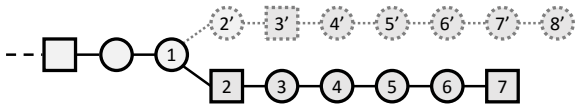

Figure 3: Key block fork. Blocks B and C have the same chain weight, and the fork is not resolved until key block D is published.

Such changes are especially problematic for small altcoins. When their value rises, they observe a rapid rise in mining power, and subsequently a drop in mining power once the difficulty rises. Then, since the difficulty is high, the remaining miners need a longer time to generate the next block, potentially orders of magnitude longer.

In Bitcoin-NG, difficulty adjustments can create a similar problem; however, it only affects key blocks. Microblocks are generated at the same constant rate. As a consequence, in case of a sudden mining power drop, Bitcoin-NG's censorship resistance is reduced, as key blocks are generated infrequently. If a malicious miner becomes a leader, it will generate microblocks until an honest leader finds a key block. Nevertheless, transaction processing continues at the same rate, in microblocks. Additionally, even until the difficulty is tuned to a correct value, the ratio of time during which malicious miners are leaders remains proportional to their mining power.

Forks When issuing microblocks at a high frequency, Bitcoin-NG observes a fork almost on every key block generation, as the previous leader keeps generating microblocks until it receives the key block (Figure 2). These forks are resolved quickly — once the new key block arrives at a node, it switches to the new leader. In comparison, when running Bitcoin at such high frequency, forks are only resolved by the heaviest chain extension rule, and since different miners may mine on different branches, branches remain extant for a longer time compared to Bitcoin-NG.

Bitcoin-NG may also experience key block forks, where multiple key blocks are generated after the same prefix of key blocks, as shown in Figure 3. This rarely happens, due to the low frequency and quick propagation of the small key blocks. The duration of such a fork may be long, lasting until the next key block. The result is therefore infrequent, but long, key block forks.

Although such long forks are undesirable, they are not dangerous. The knowledge of the fork is propagated through the network, and once it reaches the nodes, they are aware of the undetermined state. All transactions that appear only on one branch are therefore uncertain until one branch gains a lead.

Double Spending Double-spending attacks remain a vulnerability in Bitcoin-NG, though to a lesser extent than in Bitcoin.

Consider a Nakamoto blockchain and a Bitcoin-NG blockchain with the same bandwidth, where the Nakamoto block interval is the same as the key-block interval. A double-spending attacker publishes a transaction t_A, receives a service from a merchant, and publishes an alternative conflicting transaction t_B. A merchant that requires very high confidence should wait for several Nakamoto blocks, or an equivalent number of Bitcoin-NG key blocks. With lower confidence requirements, the guarantees of the protocols differ.

In Nakamoto's blockchain, blocks are infrequent, and transactions are collected by miners until they find a block. Until that time, a transaction t_A can be replaced by another transaction t_B without cost. Publication of conflicting transactions with different destinations is prohibited by the standard Bitcoin software, which also warns the user of conflicting transactions propagating in the network [30].

In contrast, in Bitcoin-NG, microblocks are frequent, and so a leader commits to a transaction by placing it in a microblock. It cannot place t_B without forming a fork and subsequently losing all of its prize from its leadership epoch via a poison transaction.

Other attacks are still possible, where a miner mines before the microblock of transaction t_A and later places a conflicting t_B. Here, the attacker loses the fees of all transactions in pruned microblocks, but this may be worthwhile since the loot from the double-spend can be arbitrarily high. An attacker can mine to prune the chain in advance, and then place a conflicting transaction, or try to prune after the fact.

Reasoning about such attacks calls for a formalization of the attacker's incentives and power. We defer formal analysis that quantifies the security guarantees of Bitcoin-NG and Nakamoto's blockchain to future work. In practice, merchants perform risk analysis to choose a strategy appropriate for their business.

6 Metrics

We now detail novel metrics by which blockchains can be evaluated. These metrics are designed to evaluate the unique properties of Nakamoto consensus.

Consensus Delay Intuitively, *consensus delay* is the time it takes for a system to reach agreement. We start by defining, for a specific execution and time, how long back nodes have to look to find a point where they agree on the state.

In a specific execution of an algorithm, given a time t and a ratio $0 < \varepsilon \leq 1$, the ε *point consensus delay* is the smallest time difference Δ such that at least $\varepsilon \cdot |\mathcal{N}|$ of the nodes at time t report the same state machine transition prefix up to time $t - \Delta$. An example for the Bitcoin protocol is illustrated in Figure 4.

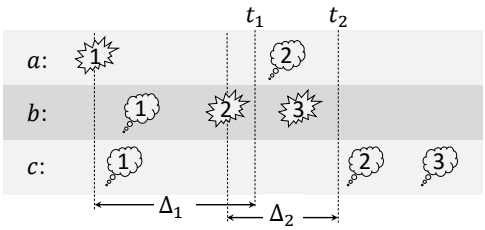

Figure 4: Point-consensus delay example with three Bitcoin nodes a, b, and c that generate blocks at heights 1, 2, and 3 (explosions) and learn that these blocks are in the main chain (clouds). Intervals Δ_1 and Δ_2 are the 50%-point consensus delays at times t_1 and t_2, respectively: At least a majority of the nodes at t_i agree on the history until $t_i - \Delta_i$.

The consensus delay is the best point-consensus-delay the system achieves for a certain fraction of the time, on average. More formally, the (ε, δ) *consensus delay* of a system is the δ-percentile ε-point-consensus-delay. For example, if 90% of the time, 50% of the nodes agree on the state of the state machine 10 seconds ago (but not less than that), then the $(50\%, 90\%)$-consensus delay is 10 seconds.

Fairness We calculate two ratios: (1) the ratio of transitions not coming from the largest miner with respect to all transitions, and (2) the ratio of mining power not owned by the largest miner with respect to all mining power. We call the ratio of these ratios the *fairness*.

Optimally the fairness is 1.0: The largest miner and the non-largest miners' representation in the transitions set should be the same as their respective mining powers.

Mining Power Utilization The security of a proof-of-work system derives from the mining power used to secure it; that is, the mining power an attacker has to outrun to obtain disproportionate control. The *mining power utilization* is the ratio between the mining power that secures the system and the total mining power. Mining power wasted on work that does not appear on the blockchain accounts for the difference.

Subjective Time to Prune Due to the probabilistic nature of Nakamoto consensus, a node may learn of a state machine transition and subsequently learn that this transition has not occurred – that it was pruned from history. This is the case with pruned branches in Bitcoin.

The δ time to prune is the δ-percentile of the difference between the time a node learns about a transition that will eventually be pruned, and the time it learns that this transition has not occurred. This implies what time a user has to wait to be confident a transition has occurred. Note that this metric only considers transitions that are eventually pruned. Figure 5 illustrates an example with the Nakamoto Blockchain.

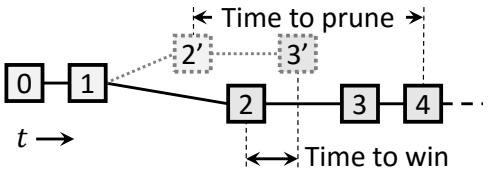

Figure 5: A fork in the blockchain with blocks drawn at their generation times, on a time X axis. *Subjective time to prune* is measured from when a node learns of a block in a branch until it realizes what the main chain is. *Time to win* is measured from the creation time of a block until the last time a node generated a conflicting block.

Time to Win The δ time to win is the δ percentile of the difference between the first time a node believes a never-to-be-pruned-transition has occurred and the last time a (different) node disagrees, believing an alternative transition has occurred. It is zero if there are no disagreements, or if the latter time is earlier. Figure 5 illustrates an example for the Bitcoin protocol.

7 Experimental Setup

We evaluate Bitcoin and Bitcoin-NG with 1000-node experiments running in real time on an emulated network.

Implementation For Bitcoin we run the standard client (release 0.10.0), hereinafter *Bitcoin*, with minimal instrumentation to log sufficient information.

We implemented all Bitcoin-NG elements that are significant for a performance analysis in the absence of an adversary, by modifying the standard Bitcoin client (release 0.10.0). We did not implement the fee distribution and the microblock signature check. Both elements have negligible impact on performance — fee distribution requires about one fixed point operation per transaction and signature checking adds several milliseconds per microblock.

Simulated Mining The time it takes a miner to find a solution follows a geometric probability distribution, which can be approximated as an exponential distribution due to the improbability of a success in each guess and the rate of guessing.

In our experiments we replace the proof of work mechanism with a scheduler that triggers block generation at different miners with exponentially distributed intervals.

Mining Power The probability of mining a block is proportional on average to the mining power used for solving the cryptopuzzle. Since blocks are generated at average set intervals and the total amount of mining power is large, the interval between block generation events of a small miner is extremely large. A single home miner using dedicated hardware is unlikely to mine a block for years [54].

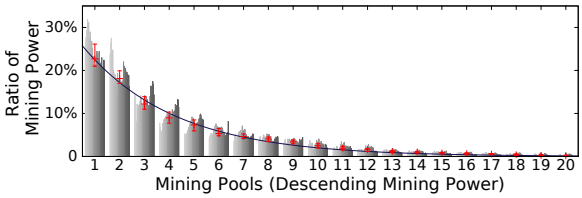

Figure 6: Error bars represent the 75th, 50th and 25th percentiles of the corresponding batch.

Consequently, mining power tends to centralize in the form of industrial mining and open mining pools. Industrial miners are companies that operate large-scale mining facilities. Smaller miners that run private mining rigs typically join forces and form mining *pools*. All members of a pool work together to mine each block, and share their revenues when one of them successfully mines a block.

To reflect in our setup the varying power of miners, we examined the hash power distribution among Bitcoin mining entities. The information we require for the analysis, the identity of the entities generating each block, is voluntarily provided by miners. We used a public API [10] to gather this information for the year ending on August 31, 2015. We note that about 9% of the blocks are unidentified. We considered each such block as generated by a different individual miner.

For each week of the year, we calculate the *weekly mining power* of each entity, and assign rank 1 to the largest weekly mining power, rank 2 to the second largest, and so on. Figure 6 shows the weekly mining power of each entity by rank up to 20. Bars of the same shade at different ranks show the distribution of a specific week. Each batch of bars represents the collection of ratios for the n^{th} highest block generating pool. We note that the ranks of different entities is not preserved throughout the weeks. The y-axis represents the weekly ratio of blocks generated by a pool.

To model the size distribution of mining entities, we approximate it with an exponential distribution with an exponent of -0.27. It yields a 0.99 coefficient of determination compared with the medians of each rank.

Network The structure of Bitcoin's overlay network is complicated, and much of it is intentionally hidden to preserve Bitcoin's security against denial of service (DoS) and to maintain participants' privacy. (Other work [29, 41] discusses details on the peer-to-peer network.) Nodes do not reveal their neighbors, but provide superset of nodes they have discovered. Many of the nodes are hidden behind firewalls making it difficult to even estimate the full size of the network. The latency among nodes is unknown. Moreover, for many of the metrics that we measure, a critical measure is the time it

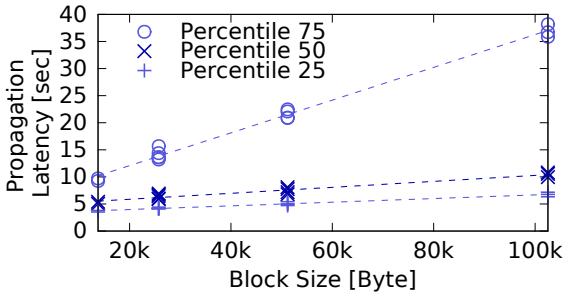

Figure 7: In our system, block propagation time grows linearly with block size. This qualitatively matches the linear relation observed in measurements of the operational Bitcoin network [18].

takes between the generation of a block by some miner and the time at which another miner starts mining on it. The block not only has to be propagated and verified by the second miner, but that second miner must also propagate the details to its mining hardware. In the case of mining pools with many distant worker miners, this may incur a non-negligible delay.

Lacking an existing model of the system, we construct a random network by connecting each node to at least 5 other nodes, chosen uniformly at random. We measured the latency to all visible Bitcoin nodes from a single vantage point on April 7th, 2015, and created a latency histogram. We then set the latency among each pair of nodes in the experiments based on this histogram. The bandwidth is set to about 100kbit/sec among each pair of nodes.

To verify the validity of our setup and topology, we compare Bitcoin's propagation properties in our setup and in the operational system. We perform experiments with different block sizes while changing the block frequency so that the transaction-per-second load is constant. Figure 7 shows a linear relation between the block size and the propagation time, similar to the linear relation measured in the Bitcoin operational network by Decker and Wattenhofer [18].

No Transaction Propagation The goal of this work is to optimize the consensus mechanism of the Blockchain. However, when generating blocks at high frequencies, the overhead of filling in the blocks by generating and propagating transactions becomes a dominant factor with Bitcoin's current implementation. This is not an inherent property of Bitcoin's protocol, or of a Blockchain protocol in general. To reduce the noise caused by the transaction generation and propagation mechanism, we reduce transaction handling to the minimum. Before starting an experiment, we initialize the blockchain with artificial transactions and top up the mempools (the data structure storing yet-to-be-serialized transactions) of all nodes with the same set of transactions. The transactions are of identical size; the operational Bitcoin system as of today, at 1MB blocks every 10 minutes, has a bandwidth of 3.5 such transactions per second.

8 Evaluation

We evaluate Bitcoin-NG and compare it with Bitcoin in two sets of experiments, varying block frequency and block size.

Overall, the experiments show that it is possible to improve Bitcoin's consensus delay and bandwidth by tuning its parameters, but its performance deteriorates dangerously on all security-related metrics. Bitcoin-NG qualitatively outperforms Bitcoin, as it suffers no such deterioration, while enjoying superior performance in almost all metrics across the entire measured range. The bandwidth of Bitcoin-NG is only limited by the processing speed of the individual nodes, as higher throughput does not introduce key-block forks. The consensus delay is determined directly by the network propagation time, because in the common case all nodes agree on the main chain once they receive the latest key block.

In the experiments that follow, we choose the 90th percentile. Lower percentiles maintain the same trends, and very low percentiles show excellent performance – there is always a small subset of nodes that has the correct chain. However, with higher percentiles, the results are lost in the noise. With 1000 nodes and at high percentiles, e.g., 99%, we are measuring the 10th slowest node. Since there are always a few nodes that lag behind, either consistently or temporarily, the results then are dominated by this random behavior, and the trends are not visible.

We measure the metrics we introduced by instantiating them to Nakamoto's blockchain and to Bitcoin-NG as follows.

Consensus delay We take the $(90\%, 90\%)$-consensus delay based on block generation times. Point-consensus-delay for Bitcoin is illustrated in Figure 4. As mentioned in Section 5, a user who requires high confidence (e.g., 99%) will not gain better latency with Bitcoin-NG, and must wait for several key blocks to accept a transaction as completed. The guarantees in such cases are similar to those of Bitcoin with the same block interval as Bitcoin-NG's key-block interval.

Fairness We calculate the proportion of (1) the ratio of blocks in the main chain not generated by the largest miner with respect to all blocks in the main chain, and (2) the ratio of blocks not generated by the largest miner with respect to all generated blocks.

Mining power utilization We calculate the proportion between the aggregate work of the main chain

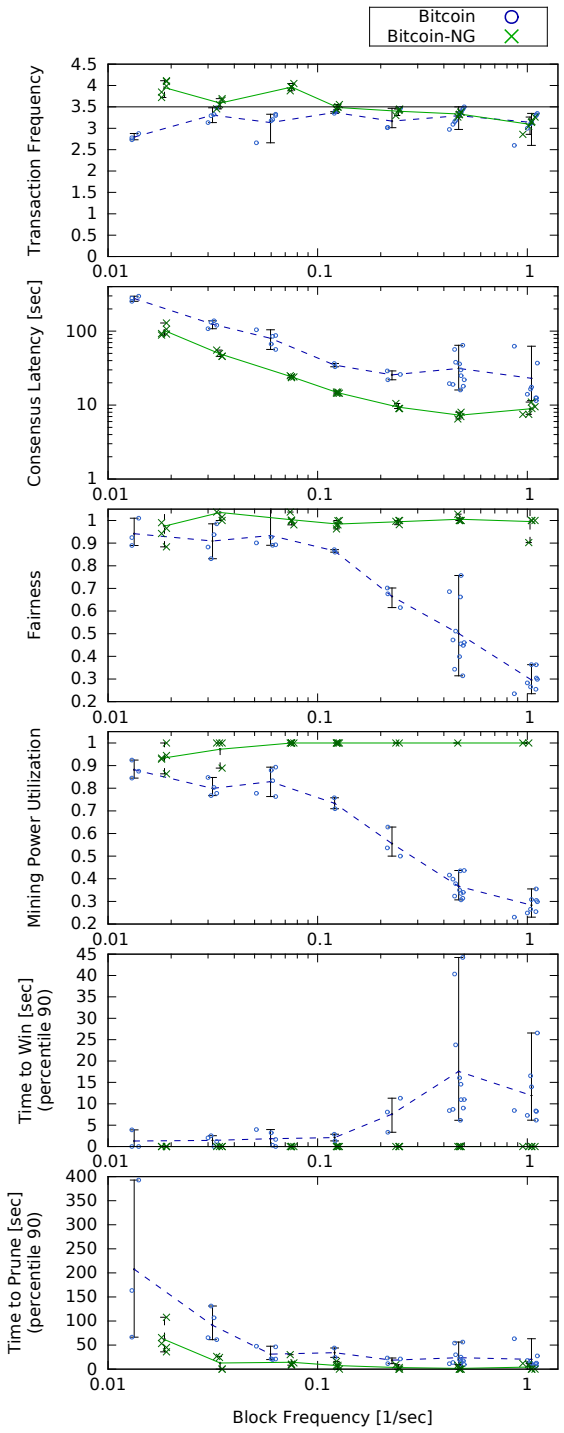

Figure 8: Reducing latency.

blocks and all blocks. In Bitcoin-NG, difficulty is only accrued in key blocks, so microblock forks do not reduce mining power utilization.

Time to prune For each node and for each branch, we measure the time it took for the node to prune this branch. This is the time between the receipt of the first branch block and the receipt of the main chain block that is longer than this branch (Figure 5). We take the 90th percentile of all samples.

Time to win We take the 90th percentile of the time from the generation of each main-chain block to the last time another miner generates a block that is not its descendant (Figure 5).

Experiments We run multiple experiments with different parameters. The figures show the average value for each group of measurements with error bars marking the extreme values. The sampled values are shown as markers.

For each execution we run for 50-100 Bitcoin blocks or Bitcoin-NG microblocks. We perform multiple short runs since all transactions are preloaded for each execution. The mean key-block interval in our experiments is 10 seconds, so each experiment includes leader changes. We do not consider cases where key-block forks occur, since in reality one would choose a much larger key-block interval, e.g., 10 minutes, making key-block forks extremely rare (more rare than with the operational Bitcoin system).

8.1 Block Frequency

First, we run experiments targeted at improving the consensus delay. For Bitcoin, we vary the frequency of block generation by reducing the proof-of-work difficulty. For Bitcoin-NG, keeping the key block generation at one every 100 seconds, we vary the frequency of microblock generation. For each frequency, we choose the block size (microblock size for Bitcoin-NG) such that the payload throughput is identical to that of Bitcoin's operational system, that is, one 1MB block every 10 minutes. Figure 8 shows the results.

We confirm that the bandwidth, measured as transaction frequency, is close to 3.5, the operational Bitcoin rate of for such transactions. In our experiments, Bitcoin's bandwidth is smaller than that of Bitcoin-NG, giving Bitcoin an advantage with respect to the other metrics.

As expected, a higher block frequency reduces Bitcoin's consensus latency as transactions are placed in the ledger at a higher frequency. Time to prune improves significantly as block frequency increases. Nevertheless, Bitcoin's frequent forks leave it with higher consensus latency and time to prune than Bitcoin-NG. We note that although they can be made arbitrarily rare, key block forks do occur. Such key-block forks are only resolved once one branch has more key blocks than the others, re-

sulting in a long time to prune if key block intervals are long.

Bitcoin's mining power utilization drops quickly as frequency increases, tending towards $1/4$, the size of the largest miner. At the extreme, block generation is so fast that by the time a miner learns of a block generated by another miner, that other miner has generated more blocks. Then, only the largest miner generates main chain blocks, and the other miners catch up. This also implies the deterioration of fairness, as forks are likely to be resolved by the largest miner extending its preferred branch. As miners struggle to catch up with the leading pack, slow miners mine on old blocks and the time to win metric increases.

Since contention in Bitcoin-NG is limited to key block generation, forks remain rare despite high frequencies of microblocks. Increasing the microblock frequency achieves reduction of both consensus delay and time to prune. All other metrics are unaffected and remain at the optimal level.

In the low-frequency experiments of Bitcoin-NG, we observe a slight mining power utilization decrease and time to prune increase. This is an artifact of the experimental setup. We run the experiments over a set number of blocks, therefore these low contention experiments run for an extended period, enough to observe key block forks. Note, however, that a realistic Bitcoin-NG implementation can space the key blocks much further apart without affecting performance. Then, due to their small size, key-block forks are highly unlikely, even more so than with standard blocks of Nakamoto's blockchain at the same rate, due to the small size of the key blocks.

8.2 Block Size

To study bandwidth scalability, we run experiments with different block sizes. We use high frequencies, similar to those of Ethereum [12], setting Bitcoin's block frequency to 1/10sec and Bitcoin-NG's microblock frequency to 1/10sec and key block frequency to 1/100sec. Figure 9 shows the result.

As expected, the transaction frequency increases with block size; the horizontal line shows the operational Bitcoin rate.

Large blocks take longer to verify and propagate. Therefore, although block frequency is constant, the time it takes for a miner to learn of a new block is longer, and so the chance for forks increases.

These experiments demonstrate the expected trade-off between bandwidth and latency. Consensus latency increases due to forks, as it takes longer to choose the main chain. The time to win also increases, as blocks take longer to catch up with the larger blocks, as does time to prune due to the many forks.

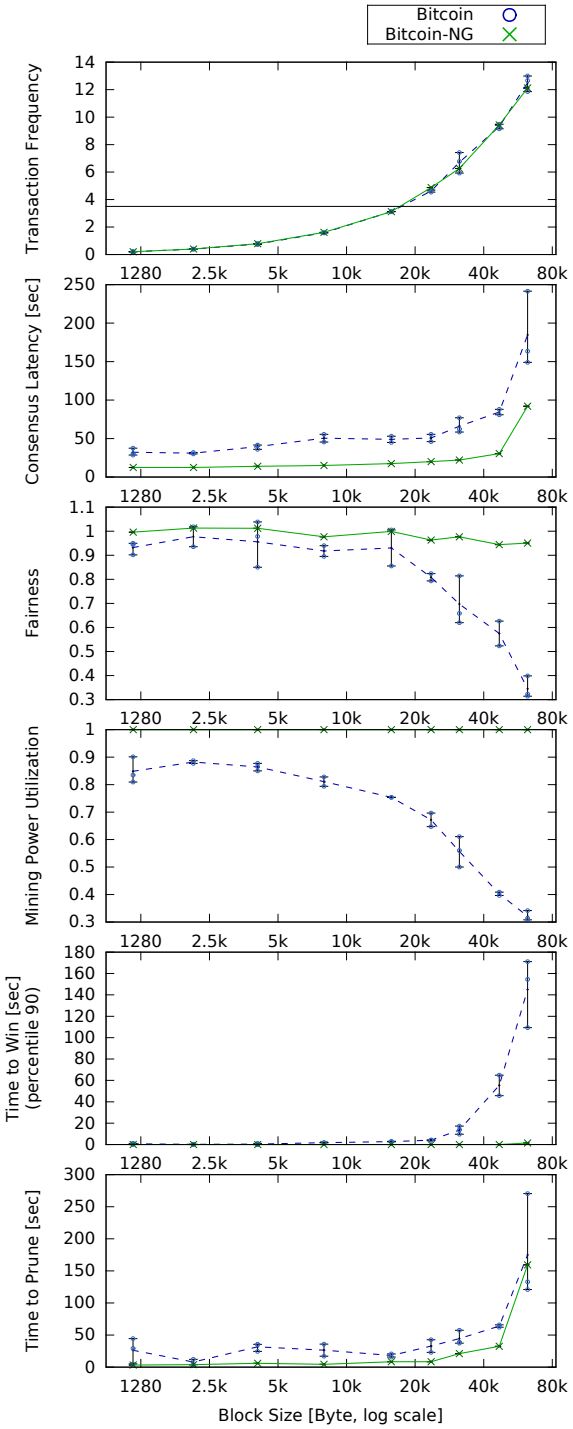

Figure 9: Increasing throughput.

While this trade-off may be acceptable, allowing for some hunt for a sweet spot on the trade-off curve, the real problem pertains to security. The forks cause significant mining power loss, reaching about 80% at Bitcoin's

bandwidth (though at a higher block frequency), making the system vulnerable to attackers that are much smaller.

Even more detrimental is the reduction in fairness. Even a minor degradation in fairness is dangerous, since it provides incentives to miners to avoid losses by joining forces to enjoy the advantage of mining in a larger pool. This leads to centralization of the mining power, obviating Bitcoin's security properties.

Bitcoin-NG demonstrates qualitative improvement, suffering no significant degradation in the security-related metrics of fairness and mining power. Under heavy load, however, the clients are approaching their processing capacity, making it hard for them to keep up, and we observe degradation in consensus latency and time to prune.

9 Related Work

Model As in Bitcoin [43] and enhancements thereof [56, 51, 36], the goal of Bitcoin-NG is to implement an RSM in an open system. The exact assumptions and guarantees are explored in different works [11, 40, 26]. Our model is similar to those of Aspnes et al. [2] and Garay et al. [26], and our definition of Nakamoto Consensus is similar to that of Garay et al. [26]. These are different from the model and goal of classical Byzantine fault tolerant RSMs. The latter, by and large, (1) assume static or slow-to-change membership, allowing for quorum systems and reconfigurations thereof, and (2) do not guarantee fairness of representation of honest parties in the state machine transitions.

The problem of leader election was apparently first formulated and solved in 1977 by Gerard LeLann [34]. In 1982, Hector Garcia-Molina addressed the problem in a distributed system that admits failures [27]. Since then leader election has been extensively used to improve the performance of distributed systems (e.g., [20, 42]). In these classical consensus protocols, the leader's role is to propose decisions that have to be confirmed by a quorum. This can be compared to blockchain protocols where the block of a leader (as defined here) is confirmed in retrospect by subsequent blocks of subsequent leaders.

GHOST The GHOST protocol of Sompolinsky et al. [51] improves on Bitcoin's scalability by changing its chain selection rule. While, in Bitcoin, the chain with the most work (accumulated over all chain blocks, based on their proofs of work) is the main chain, with GHOST, at a fork, a node chooses the side whose sub-tree contains more work (accumulated over all sub-tree blocks). The benefit is that the heaviest sub-tree choice takes into account proof of work that does not end up in the main chain. Thus, GHOST improves both fairness and the mining power utilization under high contention.

However, in GHOST, blocks on pruned subtrees only affect the selection rule at the branch point. The Bitcoin-NG protocol maintains a small fork rate at high bandwidth and throughput, allowing for better mining power utilization and fairness. Moreover, to use GHOST in an operational system, a challenge remains. In Bitcoin, at any given time, at least one node knows what the main chain is since it knows all of its blocks. In GHOST, this is not the case, and it is possible that no single node has enough information to determine which is the main chain. Our technical report [23] provides an example.

One solution to finding the true main chain in GHOST is to propagate all blocks, or all block headers [51]. However, this exposes the system to denial-of-service attacks, as a malicious node can overwhelm the network with low-difficulty blocks. There may be heuristics to avoid the security danger; we do not address this question, but have evaluated the system by implementing it, propagating all blocks. Under these conditions, GHOST performs worse than Bitcoin as the overhead of propagating all blocks outweighs the benefits of the chain selection rule. Nevertheless, a practical implementation of GHOST, overcoming remaining challenges, can be used to complement Bitcoin-NG and allow for a higher frequency of key blocks.

Inclusive Blockchains Lewenberg et al. [36] replace the blockchain structure with a directed acyclic graph. There still is a main chain, but its blocks may refer to pruned branches to include their transactions. Analysis demonstrates considerable improvement of fairness and mining power utilization. Bitcoin-NG achieves optimal fairness and mining power utilization. Using Bitcoin-NG with an inclusive blockchain to increase key block frequency may prove problematic: Decommissioned leaders could retroactively introduce transactions and have them included by the current leader. This could allow for DoS and double-spending attacks.

Faster Bitcoin Significant effort by Bitcoin's core developers is put into improving the performance of the Bitcoin client and technical aspects of its protocol. While this work can provide significant improvement and enable better scaling, it does not eliminate the inherent limitation that stems from forks forming at high rates.

Stathakopoulou et al. suggest reducing propagation delay in the Bitcoin network [53]. However, their suggestions imply significant compromises on security. First, they have nodes propagate transaction inventories before they know the actual transactions in each inventory; this allows an attacker to swamp the network at no cost by publishing transaction IDs for non-existent transactions. Second, they form a network by having nodes prefer connections with close neighbors — exactly the opposite of the current security-oriented algorithm.

Improving the efficiency of the client [1, 45, 55] can improve propagation time and reduce the collision window (time before A hears B found a block). However, the improvement is limited — a processing speed increase of $x\%$ (e.g., $x = 200\%$ with [55]) allows for block size increase of $x\%$ at the same fork rate. Bitcoin-NG provides a qualitative improvement that removes the fork rate dependency on block size or rate.

Corallo [17] has built a centralized fast relay for Bitcoin, parallel to the standard peer-to-peer network. It significantly improves network throughput and latency but increases centralized control and reduces fairness — miners outside the fast relay are at a disadvantage.

Off-chain solutions An alternative to improving the bandwidth and latency of the blockchain is to perform transactions off the chain. This basic premise apparently originated in Hearn and Spilman's two-point channel protocol [28]. The Lightning network [48] and duplex micropayment channels [19] allow for payment networks layered on top of a blockchain. The security and privacy guarantees of such payment networks differ from those of Bitcoin; as an extreme example, if the nodes performing transactions over a channel crash, all their transactions are lost, as they were never stored in the blockchain. Moreover, the efficacy of such solutions depends on properties of the emergent payment network, its topology, the amount of value locked in payment channels, as well as the protocol's ability to discover and use payment paths. Overall, these solutions may be suitable for targeted use cases where the additional layer may reduce the number of transactions seen at the lower layers, but, unlike Bitcoin-NG, they do not address the fundamental problem of scaling a Nakamoto-consensus RSM.

Another proposition for improving performance is that of federated chains, known as side chains [4], where transactions can move coins from one chain to another. Sidechains provide extensibility, as different chains can offer different features. However, their contribution for efficiency is limited, as they incur high latencies for crossing chains; moreover, when the payor has funds on one sidechain and the payee would like to spend them on another, the funds have to cross the main chain in order to get the value to their intended destination.

Analysis Given a cryptopuzzle difficulty and a topology, Sompolinsky et al. [52] calculate upper and lower bounds for the growth rate of the Bitcoin main chain. This analysis can be translated to the expected forking frequency at different difficulty levels when there are exactly two miners. Our experiments target a larger number of miners, modeled according to Bitcoin's operational system, that tune difficulty arbitrarily to reach a target main chain extension rate.

Miller and Jansen [39] describe a methodology for evaluating a large-scale Bitcoin blockchain system on a single machine using an event-driven simulator. To facilitate manageable experiment times, they replace time-consuming cryptographic operations with a delay of an appropriate length. In our experiments, we run the original operational client directly on the operating system, emulating only the network properties and mining events.

Incentives Incentive compatibility has been a key issue in the investigation of cryptocurrencies. Babaioff et al. [3] suggest a mechanism to motivate transaction propagation. Lewenberg et al. [35] propose an alternative to the chain structure to motivate the participation of badly-connected miners. Eyal [21] shows that a natural incentive system deters the formation of large open mining pools.

10 Conclusion

As Bitcoin and related cryptocurrencies have become surprisingly popular, they have hit scalability limits. The technical debate to improve scalability has been hampered by a perceived inherent trade-off between performance metrics and security goals of the system. Consequently, the discussions have become acrimonious, long-term solutions have seemed elusive, and the current sentiment has centered around short-term, incremental, compromise solutions.

Bitcoin-NG shows that it is possible to improve the scalability of blockchain protocols to the point where the consensus latency is limited solely by the network diameter and the throughput bottleneck lies only in node processing power. Such scaling is key in allowing for blockchain technology to fulfill its promise of implementing trustless consensus for a variety of demanding applications including payments, digital asset transactions, and smart contracts — at global scale.

Acknowledgments The authors thank Ayush Dubey, Gregory Maxwell, Malte Möser, Weijia Song, the paper's shepherd Jay Lorch, and the anonymous reviewers for their comments on earlier versions of this manuscript.

References

[1] ANDRESEN, G. O(1) block propagation. https://gist.github.com/gavinandresen/#file-blockpropagation-md, retrieved July. 2015.

[2] ASPNES, J. Randomized protocols for asynchronous consensus. *Distributed Computing 16*, 2-3 (2003), 165–175.

[3] BABAIOFF, M., DOBZINSKI, S., OREN, S., AND ZOHAR, A. On Bitcoin and red balloons. In *ACM Conference on Electronic Commerce* (Valencia, Spain, 2012), pp. 56–73.

[4] BACK, A., CORALLO, M., DASHJR, L., FRIEDENBACH, M., MAXWELL, G., MILLER, A., POELSTRA, A., TIMN, J., AND WUILLE, P. Enabling blockchain innovations with pegged sidechains. http://cs.umd.edu/projects/coinscope/coinscope.pdf, 2014.

[5] BAMERT, T., DECKER, C., ELSEN, L., WATTENHOFER, R., AND WELTEN, S. Have a snack, pay with Bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (2013), IEEE, pp. 1–5.

[6] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security* (1993), ACM, pp. 62–73.

[7] BITCOIN COMMUNITY. Bitcoin source. https://github.com/bitcoin/bitcoin, retrieved Mar. 2015.

[8] BITCOIN COMMUNITY. Protocol rules. https://en.bitcoin.it/wiki/Protocol_rules, retrieved Sep. 2013.

[9] BITCOIN COMMUNITY. Protocol specification. https://en.bitcoin.it/wiki/Protocol_specification, retrieved Sep. 2013.

[10] BLOCKTRAIL. BlockTrail API. https://www.blocktrail.com/api/docs#api_data, retrieved Sep. 2015.

[11] BONNEAU, J., MILLER, A., CLARK, J., NARAYANAN, A., KROLL, J. A., AND FELTEN, E. W. Research perspectives on Bitcoin and second-generation cryptocurrencies. In *Symposium on Security and Privacy* (San Jose, CA, USA, 2015), IEEE.

[12] BUTERIN, V. A next generation smart contract & decentralized application platform. https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf/, retrieved Feb. 2015, 2013.

[13] BUTERIN, V. Slasher: A punitive proof-of-stake algorithm. https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/, January 2015.

[14] CNNMONEY STAFF. The Ashley Madison hack...in 2 minutes. http://money.cnn.com/2015/08/24/technology/ashley-madison-hack-in-2-minutes/, retrieved Sep. 2015.

[15] COINDESK. Bitcoin venture capital. http://www.coindesk.com/bitcoin-venture-capital/, retrieved Sep. 2015.

[16] COLORED COINS PROJECT. Colored Coins. http://coloredcoins.org/, retrieved Sep. 2015.

[17] CORALLO, M. High-speed Bitcoin relay network. http://sourceforge.net/p/bitcoin/mailman/message/31604935/, November 2013.

[18] DECKER, C., AND WATTENHOFER, R. Information propagation in the Bitcoin network. In *IEEE P2P* (Trento, Italy, 2013).

[19] DECKER, C., AND WATTENHOFER, R. A fast and scalable payment network with Bitcoin Duplex Micropayment Channels. In *Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings* (2015), Springer, pp. 3–18.

[20] DWORK, C., LYNCH, N. A., AND STOCKMEYER, L. J. Consensus in the presence of partial synchrony. *J. ACM 35*, 2 (1988), 288–323.

[21] EYAL, I. The miner's dilemma. In *IEEE Symposium on Security and Privacy* (2015), pp. 89–103.

[22] EYAL, I., BIRMAN, K., AND VAN RENESSE, R. Cache serializability: Reducing inconsistency in edge transactions. In *35th IEEE International Conference on Distributed Computing Systems* (2015), pp. 686–695.

[23] EYAL, I., GENCER, A. E., SIRER, E. G., AND VAN RENESSE, R. Bitcoin-ng: A scalable blockchain protocol. *arXiv preprint arXiv:1510.02037* (2015).

[24] EYAL, I., AND SIRER, E. G. Bitcoin is broken. http://hackingdistributed.com/2013/11/04/bitcoin-is-broken/, 2013.

[25] EYAL, I., AND SIRER, E. G. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security* (2014).

[26] GARAY, J. A., KIAYIAS, A., AND LEONARDOS, N. The Bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), pp. 281–310.

[27] GARCIA-MOLINA, H. Elections in a distributed computing system. *Computers, IEEE Transactions on 100*, 1 (1982), 48–59.

[28] HEARN, M., AND SPILMAN, J. Rapidly-adjusted (micro)payments to a pre-determined party. https://en.bitcoin.it/wiki/Contract, retrieved Sep. 2015.

[29] HEILMAN, E., KENDLER, A., ZOHAR, A., AND GOLDBERG, S. Eclipse attacks on Bitcoin's peer-to-peer network. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.* (2015), pp. 129–144.

[30] KARAME, G. O., ANDROULAKI, E., AND CAPKUN, S. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (2012), CCS '12, ACM, pp. 906–917.

[31] KOSBA, A., MILLER, A., SHI, E., WEN, Z., AND PAPAMANTHOU, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. Cryptology ePrint Archive, Report 2015/675, 2015. http://eprint.iacr.org/.

[32] KROLL, J. A., DAVEY, I. C., AND FELTEN, E. W. The economics of Bitcoin mining or, Bitcoin in the presence of adversaries. In *Workshop on the Economics of Information Security* (2013).

[33] LAMPORT, L. Using time instead of timeout for fault-tolerant distributed systems. *ACM Transactions on Programming Languages and Systems 6*, 2 (Apr. 1984), 254–280.

[34] LE LANN, G. Distributed systems-towards a formal approach. In *IFIP Congress* (1977), vol. 7, Toronto, pp. 155–160.

[35] LEWENBERG, Y., BACHRACH, Y., SOMPOLINSKY, Y., ZOHAR, A., AND ROSENSCHEIN, J. S. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), International Foundation for Autonomous Agents and Multiagent Systems, pp. 919–927.

[36] LEWENBERG, Y., SOMPOLINSKY, Y., AND ZOHAR, A. Inclusive block chain protocols. In *Financial Cryptography* (Puerto Rico, 2015).

[37] LITECOIN PROJECT. Litecoin, open source P2P digital currency. https://litecoin.org, retrieved Nov. 2014.

[38] MEIKLEJOHN, S., POMAROLE, M., JORDAN, G., LEVCHENKO, K., MCCOY, D., VOELKER, G. M., AND SAVAGE, S. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013* (2013), pp. 127–140.

[39] MILLER, A., AND JANSEN, R. Shadow-Bitcoin: Scalable simulation via direct execution of multi-threaded applications. *IACR Cryptology ePrint Archive 2015* (2015), 469.

[40] MILLER, A., AND JR., L. J. J. Anonymous Byzantine consensus from moderately-hard puzzles: A model for Bitcoin. https://socrates1024.s3.amazonaws.com/consensus.pdf, 2009.

[41] MILLER, A., LITTON, J., PACHULSKI, A., GUPTA, N., LEVIN, D., SPRING, N., AND BHATTACHARJEE, B. Preprint: Discovering Bitcoins public topology and influential nodes. http://cs.umd.edu/projects/coinscope/coinscope.pdf, 2015.

[42] MORARU, I., ANDERSEN, D. G., AND KAMINSKY, M. Egalitarian Paxos. In *ACM Symposium on Operating Systems Principles* (2012).

[43] NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf, 2008.

[44] NAYAK, K., KUMAR, S., MILLER, A., AND SHI, E. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. *IACR Cryptology ePrint Archive 2015* (2015), 796.

[45] PAZMIÑO, J. E., AND DA SILVA RODRIGUES, C. K. Simply dividing a Bitcoin network node may reduce transaction verification time. *The SIJ Transactions on Computer Networks and Communication Engineering (CNCE) 3*, 2 (February 2015), 17–21.

[46] PEASE, M. C., SHOSTAK, R. E., AND LAMPORT, L. Reaching agreement in the presence of faults. *J. ACM 27*, 2 (1980), 228–234.

[47] PECK, M. E. Adam Back says the Bitcoin fork is a coup. http://spectrum.ieee.org/tech-talk/computing/networks/the-bitcoin-for-is-a-coup, Aug 2015.

[48] POON, J., AND DRYJA, T. The Bitcoin Lightning Network. http://lightning.network/lightning-network.pdf, February 2015. Draft 0.5.

[49] SAPIRSHTEIN, A., SOMPOLINSKY, Y., AND ZOHAR, A. Optimal selfish mining strategies in Bitcoin. *CoRR abs/1507.06183* (2015).

[50] SCHNEIDER, F. B. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys 22*, 4 (Dec. 1990), 299–319.

[51] SOMPOLINSKY, Y., AND ZOHAR, A. Accelerating Bitcoin's transaction processing. fast money grows on trees, not chains. In *Financial Cryptography* (Puerto Rico, 2015).

[52] SOMPOLINSKY, Y., AND ZOHAR, A. Secure high-rate transaction processing in Bitcoin. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers* (2015), pp. 507–527.

[53] STATHAKOPOULOU, C. A faster Bitcoin network. Tech. rep., ETH, Zürich, January 2015. Semester Thesis, supervised by C. Decker and R. Wattenhofer.

[54] SWANSON, E. Bitcoin mining calculator. http://www.alloscomp.com/bitcoin/calculator, retrieved Sep. 2013.

[55] THE BITCOIN COMMUNITY. Release notes, bitcoin 0.12.0. https://github.com/bitcoin/bitcoin/blob/0.12/doc/release-notes.md, Feb 2012.

[56] THE ETHEREUM COMMUNITY. Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper, retrieved July. 2015.

[57] WIKIPEDIA. List of cryptocurrencies. https://en.wikipedia.org/wiki/List_of_cryptocurrencies, retrieved Oct. 2013.