NC-Max: Breaking the Throughput Limit of Nakamoto Consensus

Ren Zhang[1], Dingwei Zhang[1], Quake Wang[1], Jan Xie[1], and Bart Preneel[2]
[1]Nervos
{ren, dingwei, quake, j}@nervos.org
[2]imec-COSIC, KU Leuven
bart.preneel@esat.kuleuven.be

***Abstract*—First implemented in Bitcoin, Nakamoto Consensus (NC) is the most influential consensus protocol in cryptocurrencies despite all the alternative protocols designed afterward. Nevertheless, NC is trapped with a security-performance tradeoff, largely limiting the latter. In this paper, we propose NC-Max, a consensus protocol that breaks the throughput limit and enables the full utilization of the nodes' bandwidth in confirming transactions. In particular, after identifying the mechanism through which NC's security limits its performance, we decouple transaction synchronization from confirmation with a two-step mechanism, relieving the limits on the block size and the block interval. Further, we introduce an accurate dynamic difficulty adjustment mechanism to explore the real-time network condition and to adjust the protocol's throughput accordingly. We analyze the security of NC-Max, proving that it resists selfish mining and transaction withholding attacks better than NC does. Our performance evaluation shows that NC-Max not only fully exploits the nodes' bandwidth to confirm transactions, but also shortens the overall transaction confirmation latency by at least a factor of four compared to NC without compromising security.**

I. Introduction

Bitcoin [48], the most popular digital currency, implements the *Nakamoto Consensus protocol* (NC) to help all *nodes*—network participants—to reach agreement on a *blockchain*—a chain of *blocks* containing confirmed transactions. In the last decade, NC and its variants have been implemented in more than six hundred cryptocurrencies [44] with a peak total market capitalization of several hundred billion US dollars [13]. In NC, *miners*—a special kind of nodes—compete in solving a cryptographic puzzle generated from the latest block in the blockchain and a group of new transactions. A valid puzzle solution allows the miner to broadcast a new block packing these transactions, extending the blockchain. When more than one block extends the same latest block, only one ends up in the blockchain; other blocks are *orphaned* and abandoned by the network. Miners of blockchain blocks receive *block rewards* to compensate their computing efforts. The difficulty of the cryptographic puzzle is adjusted via a *difficulty adjustment mechanism* (DAM) after every *epoch*, i.e., a fixed number of blockchain blocks, based on the estimated computing power of the last epoch.

As NC enjoys distinct advantages in its security, simplicity, and flexibility, it remains the most popular consensus protocol for cryptocurrencies a decade after its inception. Nevertheless, NC suffers from a security-performance tradeoff, which hinders it from fully utilizing the nodes' bandwidth to process

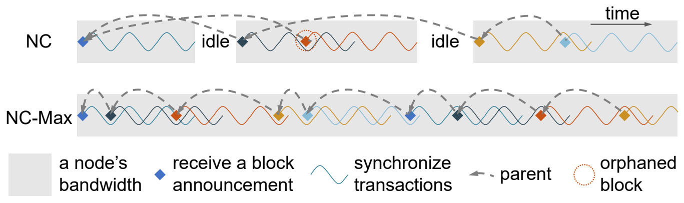


Fig. 1: NC-Max decouples transaction synchronization from confirmation to allow better bandwidth utilization and a shorter block interval without raising the orphan rate.

transactions. Specifically, NC's security demands an upper limit on the block size and a lower limit on the expected block interval, since larger blocks and shorter intervals would result in too high an *orphan rate*—the fraction of orphaned blocks, which leaves the protocol more vulnerable to various attacks [41], [65]. Such limits—e.g., Bitcoin enforces a block size limit of about 1 MB and targets a 10-minute block interval, combined with the random nature of mining, lead to insufficient utilization of the nodes' bandwidth—that the bandwidth is mostly idle between blocks, which significantly limits NC's performance potential—e.g., Bitcoin can handle merely around five transactions per second (TPS).

In this paper, we present NC-Max, a consensus protocol that raises NC's performance limit without impairing security. By introducing a two-step transaction confirmation mechanism including *transaction proposal* and *commitment*, NC-Max decouples the orphan rate from the block size, and mitigates the influence of the block interval on the former, therefore removing or alleviating the limits on these two parameters. By pipelining these two steps, NC-Max preserves the simplicity of NC's security assumptions and the reward distribution. Meanwhile, by implementing an accurate dynamic DAM, it further exploits the lower limit on the block interval, significantly reducing NC's transaction confirmation latency under the same orphan rate. Altogether, NC-Max enables the full utilization of the nodes' bandwidth without compromising security. Figure 1 presents a comparison between the two protocols' bandwidth utilization. Our contributions include:

Explicating the Linking Mechanism between the Block Size & Interval and Security. We demonstrate that the block size and the block interval determine the orphan rate by influencing the number of *fresh transactions*—transactions that have not finished propagating to the network—contained in a block. Specifically, a fresh transaction demands the nodes to request its content before forwarding the block to their peers. This extra request-and-reply round trip invalidates the *compact block* mechanism—the current block propagation acceleration technique. Through experiments deployed on, or with data collected from, the Bitcoin network, we show that increasing the block size or reducing the block interval raises the number or the proportion of fresh transactions, respectively. Synchronizing these transactions prolongs the block propagation latency, and thus also raise the orphan rate.

Relieving the Block Size & Interval Limits. Having clarified the underling linking mechanism, we introduce a two-step transaction confirmation mechanism to remove the influence of fresh transactions on the block propagation latency, and thus relieving the block size and the block interval limits set by NC's security requirements. Through introducing the two steps of *transaction proposal* and *commitment*, NC-Max pipelines the synchronization of fresh transactions and the confirmation of non-fresh transactions. Therefore, the resulting block-size-independent propagation latency reaches the lower limit permitted by the network, enabling more aggressive block size and block interval choices that more efficiently utilize the nodes' bandwidth without compromising security.

Our evaluation shows that NC-Max, under Bitcoin's current network condition and assuming a uniform bandwidth of 10 Mbps (1.25 MBps), achieves the nodes' full throughput of 2500 TPS. When anchoring the same sequence of block generation events, NC-Max's throughput outperforms recent high-throughput designs based on direct acyclic graphs (DAG) [1], [4], [42], [63], [64], since the latter waste some processing capacity on transactions that are packed multiple times in simultaneous blocks. When anchoring an orphan rate of 5%, NC-Max achieves a block interval of 5 seconds, whereas the interval must be over a minute in NC. Meanwhile, although the two-step mechanism adds a 10-block propose-commit distance to Bitcoin's six-block confirmation convention, NC-Max reduces NC's more-than-six-minute transaction confirmation latency to 80 seconds, similar to that of Prism [1]—a recent DAG protocol characterized by its short latency.

Exploiting the Performance Limit. NC-Max also introduces an accurate dynamic DAM to adjust the expected block interval, maximizing the performance under real-time network conditions. In line with previous dynamic DAMs [11], [62], we prescribe that blocks refer to not only their parents but also orphaned blocks as *uncles*, and adjust the difficulty based on the number of uncles in the last epoch. Unlike previous dynamic DAMs, which can only signal the adjusting direction—towards a longer or shorter interval, our DAM pinpoints the exact expected interval matching the network condition through delicate modeling, targeting a fixed orphan rate so that security is not compromised while the bandwidth is better exploited.

Preserving NC's Simplicity in Security Analysis and Reward Distribution. To avoid introducing new message types, NC-Max re-couples the required data of the two steps in an updated block structure. Compared to DAG protocols, this approach simplifies both the security analysis—our protocol directly inherits NC's chain growth, chain quality, and common prefix properties [26], [53]—and the reward distribution, as there is no need to share the rewards among different types of blocks or to split the fees among blocks packing the same transactions. NC-Max, therefore, inherits NC's merits and ensures most of the scientific knowledge accumulated around NC can be transferred to NC-Max.

Mitigating Selfish Mining. As one of the most fundamental attacks against NC, *selfish mining* allows attackers to gain unfair block rewards by deliberately orphaning blocks mined by other miners [24]. We prove that, through incorporating uncles in the DAM, NC-Max renders selfish mining not profitable. The proof holds regardless of the attack strategy and duration: the attacker can freely divide the mining power among honest mining, selfish mining, and idle at any time; the strategy can involve an arbitrary number of epochs.

Through this work, we aim to raise awareness on the importance of the network layer in the security and performance of blockchains. Rather than raising NC's throughput by abandoning the protocol and introducing often hard-to-solve challenges [4], [5], [38]–[42], [45], [54], [59], [63], [64], our protocol modification is built upon accurate identification and experimental verification of NC's network-layer bottleneck, which allows NC-Max to achieve better performance while preserving NC's security and flexibility. Through optimizing both the consensus and the network layers, NC-Max significantly reduces the block propagation latency and therefore fundamentally lowers the success rates of various attacks [24], [46], [47], [66], as it is more difficult for the attackers to gain any advantage in the block propagation for illegitimate purposes. We advocate that future protocol designers take a closer look at the network layer to find room for improvements.

II. The Limitations of Blockchain Consensus Protocols

This section provides a brief overview of blockchain consensus protocols and their limitations, elaborating why we choose NC as a base for further improvements.

A. Nakamoto Consensus

Protocol. NC helps all nodes agree on and order the set of confirmed transactions in a decentralized, pseudonymous way. Each block contains an 80-byte *header* in addition to the transactions. A block header includes (1) the block's *height*—distance from the hard-coded *genesis block*, (2) the hash value of the *parent*—the latest block in the blockchain, (3) the Merkle root of the transactions, (4) a timestamp, and (5) a nonce. Embedding the parent hash ensures that a miner chooses which block to mine on before starting to mine. Based on this parent-child relationship, all blocks form a tree, and each root-to-leaf path in the tree is called a *chain*. The transactions must not conflict with those in previous blocks of the same chain. To construct a valid block, miners work on finding the right nonce so that the block hash is smaller than a *target* T, which is computed by the last iteration of the DAM. Compliant miners publish blocks the moment they are found. Miners are incentivized by two kinds of rewards. First, a *block reward* is allocated to the miner of every blockchain block via a special *coinbase transaction* in the block. Second, the value difference between the inputs and the outputs in a transaction is called the *transaction fee*, which goes to the miner who includes the transaction in the blockchain.

To adjust the target T based on the network hash rate, a DAM is triggered after every epoch of M blockchain blocks:

$$T_{i+1} = \begin{cases} T_i \cdot \frac{1}{\tau}, & L_i < L_{\text{ideal}} \cdot \frac{1}{\tau} \\ T_i \cdot \tau, & L_i > L_{\text{ideal}} \cdot \tau \\ T_i \cdot \frac{L_i}{L_{\text{ideal}}}, & \text{otherwise} \end{cases}, \qquad (1)$$

where i is the epoch number, L_{ideal} is the ideal time duration of an epoch, τ is a dampening filter to prevent rapid changes of T, and L_i is the actual time duration of the last epoch, as reported in the blocks. In Bitcoin, $M = 2016$, $\tau = 4$, and L_{ideal} is two weeks so that the average block interval is ten minutes if the mining power remains constant.

When more than one block extends the same parent, miners work on the *main chain* that is most computationally challenging to produce, which is sometimes inaccurately referred to as the *longest chain*. When several chains are of the same "length", miners choose the first chain they receive. We refer to this *forked* situation where miners work on different parents as a *block race*, an equal-length block race as a *tie*, and blocks of the same height as *competing blocks*. Blocks not in the main chain are orphaned and discarded by all miners.

Limitations. A high orphan rate could negatively impact NC's security and performance [65]. As orphaned blocks do not contribute to the main chain's total proof-of-work (PoW), the more compliant miners' blocks orphaned, the less adversarial mining power it requires to secretly generate a longer chain, downgrading the system's security threshold. Meanwhile, although orphaned blocks do not contribute to the transaction confirmation, their propagation consumes the nodes' bandwidth, thus limiting the system's throughput, as this bandwidth could have been utilized to synchronize transactions.

Natural forks occur when competing blocks are discovered in an interval shorter than the propagation latency. Unfortunately, two intuitive modifications to improve NC's throughput: increasing the block size and reducing the block interval, would both raise the orphan rate. Whether to raise the throughput at the risk of downgrading security is the most heatedly-debated topic in the Bitcoin community in recent years.

Forks can also be caused by attacks. In the selfish mining attack first described by Eyal and Sirer [24], the malicious miner keeps discovered blocks secret and mines on top of them, hoping to gain a larger lead on the public chain of *honest blocks* mined by the compliant miners. The selfish miner publishes the secret chain if it has one block and the public chain catches up, or it has more than one block and the lead is reduced to one. By invalidating honest blocks, the attacker gains relative block rewards larger than his fair share.

At last, we note that faster propagation of honest blocks can not only reduce natural orphans but also discourage attacks. In an attack-induced tie, due to the first-received policy, the honest blocks' propagation latency determines the amount of compliant mining power that works on the attacker chain, and therefore determines the attack's success rate. In extreme cases, if honest blocks always propagate faster, an attacker with less than 33% of the total mining power cannot gain unfair profit from attempting selfish mining; however, if the attacker's blocks always propagate faster, the attack succeeds with any mining power share [24]. Faster propagation of honest blocks also discourages double-spending [66] and feather-forking attacks [47].

B. Innovative Consensus Protocols: Into the Unknown

Inspired by the success of NC, a considerable number of alternative consensus protocols have emerged. These efforts can be categorized into two groups: NC's variants and innovative protocols. Nevertheless, while the numerous variants of NC suffer from the same limitations as NC's and none of them comprehensively outperforms NC [76], innovative protocols all lead to new and unsolved challenges in their security, performance, or functionality. We now briefly introduce the limitations of existing innovative protocols, which are further divided into two approaches.

Proof-of-Possession Protocols. A series of *proof-of-stake* (e.g., Algorand [28], Ouroboros Praos [18], and Snow White [17]) and *proof-of-space* (e.g., Spacemint [52]) protocols select participants to compose blocks and to authorize transactions based on their possession of some scarce resources, to avoid the energy consumption in NC.

Since the resources are not consumed during the block generation process, extra mechanisms or security assumptions must be in place to prevent attackers from constructing alternative history versions. For example, Algorand demands that each block, regardless of the block size, be accompanied by a certificate of 300 KB, comprised of hundreds of digital signatures [28]. Broadcasting these signatures negatively affects performance. Moreover, these systems usually rely on trusted parties or stronger-than-NC synchrony and online assumptions, which might lead to new attack vectors when they are not met. Detailed discussions on the limitations of proof-of-possession protocols can be found in [3], [8].

DAG Protocols. SPECTRE [63], Meshcash [4], PHANTOM, GHOSTDAG [64], Prism [1], and Conflux [42] suggest replacing the linear blockchain structure with a *blockDAG*—a direct acyclic graph of blocks. In a DAG protocol, rather than referring to a single parent, a block contains hashes to all blocks the miner has received satisfying certain conditions. By confirming transactions with blocks not necessarily in a chain, these protocols hope to achieve higher throughput than chain-based protocols.

DAG protocols' actual throughput is yet to be quantified, as it is difficult to model how much bandwidth is wasted due to transactions embedded multiple times in simultaneous blocks. This *duplicate packing problem* further complicates the transaction fee distribution, whose mechanism is omitted in PHANTOM, GHOSTDAG [64] and Prism [1]. In particular, Prism, with its three kinds of blocks, also introduces potential incentive issues in block reward distribution and new attacks against its DAM, which have not been accounted for in its design. At last, as the global order of transactions is not known when constructing a block, transaction validity can only be evaluated after the neighboring blockDAG topology is settled, resulting in a long confirmation delay. Alternatively, SPECTRE abandons the global order, rendering it incompatible with the most popular smart contract programming model [10].

C. NC as a Base for Future Endeavors

Despite the emergence of numerous alternatives, NC still has a threefold advantage compared to its alternatives. First, it is built on only a minimum set of security assumptions. Consequently, NC's security is carefully scrutinized and well-understood [21], [24]–[26], [35], [53], [61], [72]. Alternative protocols often open new attack vectors, either unintentionally [34], [76] or by relying on security assumptions that are difficult to realize [3], [8]. Second, NC minimizes the consensus protocol's communication overhead. In the best-case scenario, propagating a 1 MB block in Bitcoin is equivalent to broadcasting a compact block message of 13 KB [14], [46]; valid blocks are immediately accepted by all honest nodes. In contrast, alternative protocols often demand a non-negligible communication overhead to certify that certain nodes have witnessed a block or to transmit the same transactions multiple times. Third, unlike DAG protocols, NC's chain-based topology ensures that a transaction global order is determined at block generation, which (1) minimizes the transaction verification and confirmation latencies, and (2) is compatible with all smart contract programming models [10], [55].

Therefore, we argue that NC provides the most promising base for future protocol designs. Nevertheless, NC's unsatisfactory performance and, in particular, the unsuccessful attempts of NC's variants despite all the scholarly and engineering efforts signal some fundamental tension in NC that limits its performance. We will unpack this tension in the next section.

III. NC's Performance Bottleneck

Given the limitations of NC, the Bitcoin developers introduced *compact blocks* (CB) in 2016 to lower the orphan rate by reducing the block propagation latency. However, with CB implemented, fresh transactions become the obstacle in lowering the block interval or increasing the block size. This section first introduces CB, and then elaborates on how fresh transactions defer the block propagation and enable a transaction withholding attack, followed by experiments that confirm the identified bottleneck.

A. Compact Blocks

We briefly overview the protocol here and refer interested readers to [14] for the specification. CB reduces the block propagation latency by optimistically not transferring full transactions in a block by default. Each node supporting CB enables the *HB mode* with the last three peers that have sent blocks to the node. Whenever a new block is available at an HB peer, the peer constructs a CB by replacing each transaction with a 48-bit shortid, and adding two extra fields: (1) a connection-specific 64-bit random salt and (2) a list of transactions for which the peer is certain that the receiving node is not aware of them—just the coinbase transaction in the current implementation. The second field is called *prefilled transactions*. A shortid is computed as siphash-2-4(txid,SHA-256(*header*||*salt*)), where siphash-2-4 and SHA-256 are two hash functions, txid is the transaction hash, and *header* contains the block's metadata. Replacing transactions of typically two to five hundred bytes each with their shortids can compress a 1 MB block into around 13 KB [46], which is significantly faster to transfer. The CB is then sent to the node. After receiving the CB, the node computes shortids for transactions in its unconfirmed transaction pool, matches them with those in the CB, and requests the missing transactions. Once the node receives these transactions, it constructs and forwards relevant CBs to its other peers, and starts verifying the newly received transactions in the meantime.

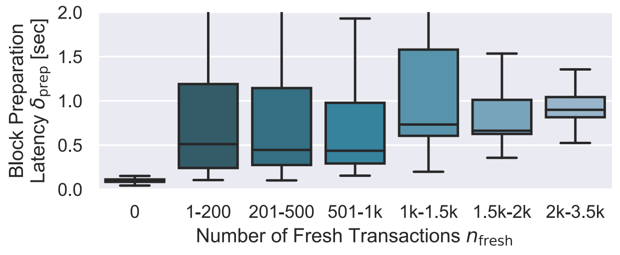


Fig. 2: The block preparation latency δ_{prep} is higher when a block contains fresh transactions, i.e., $n_{\text{fresh}} > 0$. The ends of a box are the 25th and 75th percentiles of our data set; a horizontal line inside the box marks the median. The maximum latency is not displayed if it is over two seconds.

B. Fresh Transactions and Transaction Withholding Attacks

According to the Bitcoin developers, after implementing CB, the main contributor to the block propagation latency is the extra round trip caused by the missing transactions, if there is one [46]. Not all nodes have stored these *fresh transactions* in their memory pools, as these transactions are usually broadcast only a few seconds before they are mined in a block. This extra-round-trip delay is magnified as it usually takes several hops for the block to reach the entire network.

As of 2020, the time of this writing, most Bitcoin blocks contain no fresh transactions thanks to the ten-minute block interval. Since it takes only 15 seconds for a transaction to reach 90% of nodes [20], fresh transactions constitute only a small fraction of the transactions broadcast after the last block. This fraction increases if the block interval is shortened.

Furthermore, resourceful miners can raise their revenues by deliberately packing transactions only known to themselves, which we termed *transaction withholding attacks*. The slower block propagation gives them more time to mine on their blocks before other miners have received them, hoping to orphan honest blocks mined during this period as they do not extend the longest chain, thus constituting a de facto selfish mining attack. As selfish mining's profitability raises superlinearly with the attacker's mining power, more resourceful miners have less incentive to accelerate their blocks' propagation [24], [46]. This attack differs from selfish mining in that the attacker does not delay his blocks further than their natural propagation latency. We will analyze this attack and NC-Max's resistance against it in Sect. VI-C.

C. Confirming the Bottleneck

Fresh Transactions Affect the Block Propagation Latency. We first study the relation between the number of fresh transactions, denoted as n_{fresh}, and the one-hop block propagation latency δ. The multi-hop latency is studied in Sect. VII.

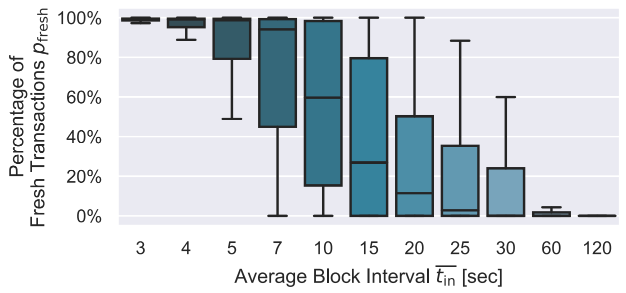


Fig. 3: The percentage of fresh transactions in a block p_{fresh} increases when the block interval $\overline{t_{\text{in}}}$ decreases.

The one-hop latency can be further decomposed into two parts: $\delta = \delta_{\text{CB}} + \delta_{\text{prep}}$, where δ_{CB} denotes *the CB transmission latency* from a node's upstream peer to the node and δ_{prep} denotes *the block preparation latency*, i.e., the time between the node's receiving a CB and forwarding CBs to its peers. As CBs are small, δ_{CB} equals the latency between the upstream peer and the node, typically 1 to 50 ms [20], as upstream peers are usually a node's most well-connected peers. The extra round trip to request missing transactions is included in δ_{prep}.

As δ_{CB} is relatively short and independent of n_{fresh}, we focus on the relation between n_{fresh} and δ_{prep}. We modified the Bitcoin client v0.17.1 to reject a fraction of new transactions to increase n_{fresh} when receiving a new block, and to log n_{fresh} and δ_{prep} for every block it receives. We deployed three instances of the modified client on IP addresses 47.251.4.119, 47.244.165.26, 8.208.203.101, located in the US, Hong Kong, and the UK, respectively, and collected data from the Bitcoin network between May 10th and May 18th, 2019.

As shown in Fig. 2, when there is no fresh transaction, δ_{prep} is short and stable, typically around 100 ms. In this case, the latency is the time to reconstruct the block from the CB and to generate CBs for its peers. When $n_{\text{fresh}} > 0$, even if the number is small, the median of δ_{prep} raises to over 500 ms, due to the request-and-reply round trip. The latency becomes even higher when there are more than 1000 fresh transactions, as larger messages take longer to transmit.

The Fraction of Fresh Transactions Increases When the Block Interval Decreases. Next, we study the relation between the expected block interval, denoted as $\overline{t_{\text{in}}}$, and the fraction of fresh transactions, denoted as p_{fresh}.

We simulate a network of 6000 nodes. In line with Bitcoin, each node establishes eight outgoing connections. Transactions and blocks are gossiped to the network from their initiators, with a fixed one-hop latency of 3.59 and 1.20 seconds, respectively, so that the total propagation time—usually five hops—is similar to those of Bitcoin's, i.e., roughly 15 and 5 seconds, respectively [20]. A hundred transactions are generated per second, each from a random node as its initiator. A transaction is simulated as a dummy message with an ID and a timestamp. Each node maintains a memory pool of unconfirmed transactions. Each block is also generated by a random node. The block interval follows an exponential distribution with an average value of $\overline{t_{\text{in}}}$ as the input. A miner packs up to $100\overline{t_{\text{in}}}$ transactions from its memory pool, prioritizing those with the oldest timestamps, as the block. "Packing from the oldest" results in a lower p_{fresh} than in reality, where miners pack from the highest fee-per-byte transaction. We execute 100 iterations with 100 blocks each for every $\overline{t_{\text{in}}}$. Each block's p_{fresh} is averaged over all the receiving nodes.

The results are displayed in Fig. 3. There are very few fresh transactions in blocks when $\overline{t_{\text{in}}} = 120$ seconds, which is consistent with Bitcoin's current situation. However, p_{fresh} increases when the block interval decreases, especially when it drops below the total transaction propagation latency.

Combining these results, we conclude that, when reducing NC's block interval, there will be more fresh transactions in the blocks, which increase the block propagation latency. Previous studies [19], [65] demonstrate that increasing the block size also raises this latency as more transactions take longer to synchronize, which will be further confirmed in Sect VII. A higher block propagation latency directly results in a higher orphan rate, which is undesirable for NC's security. Below, we present our solution to this performance bottleneck.

IV. NC-Max: Two-Step Confirmation

To decrease the block propagation latency and thus increasing the throughput, NC-Max introduces two new mechanisms: (1) two-step transaction confirmation, which is described in this section, and (2) a dynamic DAM, described in Sect. V. The two-step mechanism introduces two adjustments to NC:

Incorporating Uncles. Miners are requested to refer to *uncles*—orphaned blocks in the same epoch—by embedding their hashes in the blocks. Our uncle's definition differs from that of Ethereum [10], the cryptocurrency with the second-largest market capitalization, in that our uncle's validity does not consider how far away the uncle and the nephew's first common ancestor is. The reward distribution regarding uncles is also different from that of Ethereum, which we elaborate in Sect. IV-D.

Prescribing a Transaction Proposal Step. A *transaction proposal zone* is added to each block, containing txpids—the first several bytes of transaction hashes—of some possibly fresh transactions. We consider these transactions *proposed* in this block. Transactions proposed in uncles are also considered proposed in the nephew so that uncles also contribute to the transaction proposal. The set of full transactions in a block is now called the *transaction commitment zone*. The proposed transactions—unlike their txpids—are not part of the block, therefore they do not affect the block's validity: a block may still be valid if some txpids refer to malformed or double-spending transactions, or the miner refuses to provide the full content of some newly proposed transactions. The uncles' proposal zones are part of the block, and optionally part of the CBs if they have not been synchronized in the current connection. The most important rule about the proposal zone is that a transaction in the commitment zone of a block with height h must be proposed in a main chain block whose height is between $h - w_{\text{far}}$ and $h - w_{\text{close}}$, a range we termed *the proposal window*.

As proposed transactions do not affect a block's validity, a node forwards the CBs—including the full proposal zone—to its downstream peers as soon as it finishes reconstructing the commitment zone, whose transactions are already synchronized after receiving the blocks in the proposal window. The

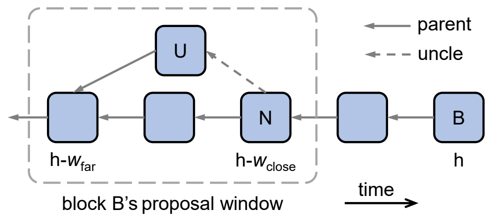


Fig. 4: Block B can only commit transactions proposed in its proposal window. In this example, $w_{\text{close}} = 2$, $w_{\text{far}} = 4$. If a transaction is only proposed in U and committed in B, its proposal fee goes to N's miner.

request to the node's upstream peers for missing proposed transactions in the latest block are sent in the meantime. Consequently, the round-trip time of requesting the missing transactions is removed from the critical path of block propagation, reducing the propagation latency. Malicious miners may still conduct the transaction withholding attack by refusing to provide the full transactions they proposed and then hoping to commit these secret transactions in the future; however, the success rate and the damage of this attack are lower in NC-Max than in NC, as we will show in Sect. VI-C.

Next, we formally define the two steps and the block structure, and then introduce the new block propagation protocol and our reward distribution mechanism.

A. Definitions

Definition 1 (Proposal id)**.** *A transaction's* proposal id txpid *is defined as the first l bits of the transaction hash* txid.

A txpid does not need to be globally unique as the 32-byte txid, as a txpid is used to identify a transaction among several neighboring blocks. Since we embed txpids in both blocks and CBs, sending only the truncated txids could reduce the bandwidth consumption. When multiple transactions share the same txpids, all of them are considered proposed. In practice, we can set l to be large enough so that the computational effort of finding a collision is non-trivial. For example, when $l = 144$, finding a txpid collision is roughly as difficult as mining a block in Bitcoin.

Definition 2 (Uncle)**.** *A block B_1 is considered an* uncle *of another block B_2 if all of the following conditions are met: (1) they are in the same epoch, sharing the same difficulty; (2) $height(B_2) > height(B_1)$; (3) B_1's parent is already embedded in B_2's chain, either as a main chain block or as an uncle; and (4) B_2 is the first block in its chain to refer to B_1.*

Condition (1) enables uncles to contribute to a more accurate hash rate estimation, which will be exploited in our DAM. A violation of (2) contradicts the longest-chain rule. Condition (3) is to ensure that two NC-Max instances with different genesis blocks do not accidentally recognize each other's blocks as uncles. An example can be found in Fig. 4.

Definition 3 (Transaction proposal)**.** *A transaction is* proposed *at height h_p if its* txpid *is in the* proposal zones *of the main chain block with height h_p or of this block's uncles.*

The proposal zone is used to facilitate transaction synchronization. The proposed transactions' validity does not affect the block's validity.

Definition 4 (Transaction commitment)**.** *A non-coinbase transaction is* committed *at height h_c if all of the following conditions are met: (1) it is proposed at height h_p of the same chain, where $w_{\text{close}} \leq h_c - h_p \leq w_{\text{far}}$; (2) it is in the* commitment zone *of the main chain block with height h_c; and (3) it is not in conflict with any previously-committed transactions in the main chain. The coinbase transaction is committed at height h_c if it satisfies (2) and (3).*

We borrow the term *commit* from the literature for convenience, despite that in line with NC, NC-Max only offers probabilistic confirmation. Parameters w_{close} and w_{far} define the proposal window—the closest and farthest on-chain distance between a transaction's proposal and commitment, as shown in Fig. 4. We require w_{close} be: (1) more than one, to defend against the transaction withholding attack; (2) large enough, so that w_{close} block intervals are long enough for newly proposed transactions to be propagated to the network. The selection of w_{close} will be further discussed in Sect. VII-D. Enforcing a proposal window also guarantees that the "proposed transaction pool" fits in a node's memory. A transaction is considered embedded in the blockchain when it is committed.

B. Block and Compact Block Structure

Data Structure. A block includes the following fields:

header	block metadata
commitment zone	full transactions
proposal zone	txpids
uncle headers	headers of the uncles
uncles' proposal zones	txpids in the uncles

The header contains the Merkle roots of the commitment zone, the proposal zone, and uncle headers, in addition to NC's header. Similar to NC, a CB replaces the commitment zone with the transactions' shortids, a salt, and a list of prefilled transactions. All other fields remain unchanged in the CB.

A *block size limit* is applied to the total size of the first four fields, to limit the data size that needs to be synchronized across the network along with each PoW solution. The uncles' proposal zones do not count in this limit as they are usually synchronized before the block is mined. The number of txpids in a proposal zone also has a hard-coded upper bound.

Parameter Recommendation. Two heuristic requirements can help practitioners choose the parameters. First, the upper bound on the number of txpids in a proposal zone should be no smaller than the maximum number of committed transactions, so that even if $w_{\text{close}} = w_{\text{far}}$, this bound is not the protocol's throughput bottleneck. Second, ideally, a CB should be no bigger than 80 KB. According to Croman et al. [16], messages no larger than 80 KB have similar propagation latency in the Bitcoin network in 2016; larger messages propagate slower as the nodes' bandwidth becomes the bottleneck. This number changes as the network condition improves. A typical set of parameters can be found in Appendix A.

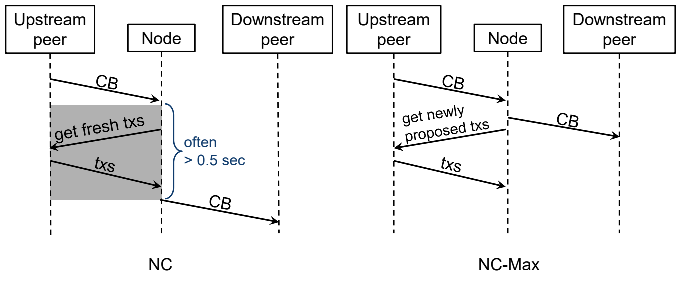


Fig. 5: Block propagation in NC and NC-Max. "Transactions" is abbreviated as "txs". In most occasions, our protocol allows nodes to forward CBs to their peers as soon as they received them, as all committed transactions are already synchronized.

C. Block Propagation Protocol

In line with [2], [24], [32], nodes should broadcast all blocks with valid PoWs, including orphans, as they may become uncles. Valid PoWs cannot be utilized to pollute the network, as constructing them is energy-consuming.

On most occasions, NC-Max's block propagation protocol removes the extra round trip of fresh transactions, as shown in Fig. 5, so that blocks are propagated with a constant latency regardless of how many transactions are proposed; when the round trip is inevitable, NC-Max ensures that it only lasts for one hop in the propagation and the additional latency is limited. This is achieved by the following three rules.

R1: non-blocking transaction query. As soon as the commitment zone is reconstructed, a node forwards the CBs to its downstream peers and queries the newly-proposed transactions from its upstream peers simultaneously.

The block propagation will not be affected by these transaction queries as long as they are answered before the next w_{close}-th block is mined.

R2: missing transactions, now or never. If certain committed transactions are unknown to a CB receiver, the receiver queries the sender with a short timeout. Failure to send these transactions in time leads to the receiver disconnecting the sender. If the disconnected sender was an outgoing connection, the receiver establishes a new connection to a random node. Moreover, the incomplete block will not be propagated further before receiving these transactions from another peer.

Proposed-but-not-received transactions are committed either (1) in a successful transaction withholding attack, or (2) when w_{close} consecutive blocks are mined before the transactions proposed in the first one are synchronized. If the upstream peer is honest, as in (2), a short timeout is adequate to transfer the missing transactions, as honest nodes will not send CBs until they finish reconstructing the block. In the case of (1), the attacker cannot delay the first hop of the block propagation more than the timeout value without the block being discarded. In practice, we set the timeout to be 2 seconds.

R3: transaction push. If certain committed transactions are previously unknown to a CB sender, they will be embedded in the prefilled transaction list and sent along with the CB.

This rule removes the round trip if the sender and the receiver share the same list of proposed-but-not-broadcast transactions. In a transaction withholding attack, this rule ensures that the secret transactions are only queried in the first hop of the block's propagation, and then pushed directly to the receivers in subsequent hops.

D. Reward Allocation

A fixed block reward goes to every main chain block miner, which is calculated based on Eqn. (10) in Appendix B-C. For each committed transaction, 60% of its transaction fee, denoted as the *commitment fee*, goes to the main chain block miner who commits it; while the other 40%, denoted as the *proposal fee*, goes to the earliest main chain block miner who proposes the transaction within the proposal window. This 60-to-40 fee allocation balances the miners' incentives to propose transactions and to extend the longest chain, in line with Bitcoin-NG [23]. Uncle miners do not get any fees.

NC-Max—unlike Ethereum—issues neither uncle rewards to compensate uncle miners, nor nephew rewards to incentivize miners to embed uncles in their blocks. This is because uncle and nephew rewards raise the selfish mining profit and lower the mining power threshold to perform the attack [51], [58].

Miners embed uncles for three kinds of benefits. First, embedding uncles allows more transactions to be proposed in a block, thus earning more proposal fees for the nephew miner. Second, when there are not enough transactions to propose in a nephew block, some transactions can be omitted if they are already proposed in the uncles, to further accelerate the nephew's propagation and lower its probability of being orphaned. Third, embedding uncles contributes to a more accurate estimation of the network hash rate, thus contributing to the system's selfish mining resistance (Sect. VI-D).

V. NC-Max: Dynamic DAM

NC-Max further exploits the bandwidth utilization to the limit of the real-time network condition with an accurate dynamic DAM. Our goal is twofold: (**G1**) to render selfish mining unprofitable; (**G2**) to dynamically adjust the throughput based on the network's bandwidth and latency. Meanwhile, to maximize compatibility and attack resistance, our DAM needs to satisfy four constraints, in line with NC:

C1. All epochs have the same target duration L_{ideal}.

C2. The maximum block reward issued in an epoch $R(i)$ depends only on the epoch number i so that the rewards are distributed at a predetermined rate.

C3. The hash rate estimation of the last epoch does not change too fast, to prevent attackers from manipulating the DAM and forging a blockchain [2], even if some miners' network is temporarily controlled by the attacker.

C4. The expected block interval should abide by predetermined upper and lower bounds. The upper bound guarantees the service availability; the lower bound guarantees that NC-Max does not generate more traffic than most nodes' capacity, ensuring decentralization.

To achieve **G1**, NC-Max incorporates all blocks, instead of only the main chain, in calculating the *hash rate estimation* of the last epoch, and then applies a dampening factor to the estimation so that the adjusted output conforms to **C3**. This output determines the computing efforts required in the next epoch for each reward unit. The effectiveness of this mechanism will be proved in Sect. VI-D. To achieve **G2**, our DAM targets a fixed orphan rate, rather than a fixed block interval as in NC. As our two-step confirmation ensures a relatively stable block propagation process, we can solve the expected block interval matching the target orphan rate with the last epoch's duration, orphan rate, and the main chain block number. As the target epoch duration is fixed (**C1**), we can solve the next epoch's main chain block number, block reward and difficulty target after applying several dampening factors and upper/lower bounds to safeguard **C2** and **C4**. We defer the detailed description of our DAM to Appendix B.

VI. Security Analysis

Having introduced the core design, next, we analyze the protocol's core security metrics, its resistance against transaction withholding and selfish mining attacks.

A. Threat Model

In line with prior studies [2], [24], [27], [32], [61], [74]–[76], we assume the total mining power remains unchanged throughout the attack, and the adversarial mining power share α stays below half of the total mining power. Under this environment, we assume one strong attacker who coordinates all adversarial mining power, which can cause more damage than multiple attackers acting separately [24]. The attacker may temporarily turn off some mining equipment, hoping to manipulate the DAM for higher overall profit. The attacker receives and broadcasts messages with zero propagation delay, and can arbitrarily reorder messages. However, the attacker cannot slow down honest blocks' propagation. Other miners abide by the protocol. We do not consider the effect of transaction fees [12], [43], [69], as it only makes up 1% of the miners' total rewards in Bitcoin [6]. Neither do we consider eclipse attacks [33], [49] or network partitions.

B. The Applicability of NC's Security Proofs to NC-Max

NC-Max adopts the same backbone protocol as with NC, therefore the security proofs of NC are all directly applicable to our protocol. In particular, NC's security is specified as the upper and lower bounds on the chain quality, chain growth, and common prefix metrics, first defined and proved by Garay et al. [26]. Subsequent studies [21], [25], [35], [53], [72] optimized these bounds and loosened the assumptions. All these proofs are built solely on Bitcoin's *backbone protocol*, which NC-Max inherits with no modification.

Our modifications to NC are limited to (1) the block validity rules—specified as the *content validation predicate* and the *input contribution function* in [26]—and (2) the DAM. NC's security proofs make only two assumptions on the block validity rules, both satisfied by NC-Max: each block introduces enough entropy; a block mined by an honest miner is valid to the others. As for the DAM, the security proofs require that the block propagation latency be shorter than the block interval, which is guaranteed by our block interval lower bound (**C4**) and strengthened by the two-step confirmation. In sum, NC-Max is an instantiation of the Bitcoin backbone protocol, and thus is compatible with its potential security updates.

C. Resistance Against Transaction Withholding attacks

Success Rate. We compare the fraction of *attacker blocks*—blocks mined by the attacker—that can be slowed down in NC and NC-Max, denoted as F_{slow}, assuming all the blocks are in the main chain. Although the numerical results do not reflect the actual attack resistance, this simplification does not affect the comparison: the more blocks the attacker can slow down, the more honest blocks he can orphan.

In NC, this attack can be performed with every attacker block, namely $F_{\text{slow}}^{\text{NC}} = \alpha$. In NC-Max, a block's propagation can only be slowed down if it contains proposed-but-not-broadcast transactions. To trigger this case, the attacker needs two blocks not too far from each other in the chain: the older one to propose these secret transactions, and the younger one to commit. For every attacker block with height h, it can commit secret transactions only if there is another attacker block between $h - w_{\text{close}}$ and $h - w_{\text{far}}$. The probability that there is such a block is $1 - (1-\alpha)^{w_{\text{far}} - w_{\text{close}} + 1}$. Therefore, $F_{\text{slow}}^{\text{NC-Max}} = \alpha(1-(1-\alpha)^{w_{\text{far}} - w_{\text{close}} + 1})$. We immediately have $F_{\text{slow}}^{\text{NC}} > F_{\text{slow}}^{\text{NC-Max}}$, namely, NC-Max offers stronger resistance to transaction withholding attacks than NC.

Simulation. We simulate both protocols to demonstrate NC-Max's resistance against this attack. Our simulation can model how α, the proposal window size w, the maximum attack-incurred latency, and orphaned blocks affect the attacker's unfair percentage of main chain blocks. Our results show that when $\alpha = 0.4$, NC-Max reduces the unfair percentage by roughly a half and a third when $w = 1$ and 2, respectively. The detailed model and our results can be found in Appendix C.

D. Selfish Mining Resistance

We prove the following theorem on the selfish miner's profitability in Appendix D.

Theorem 1 (Selfish mining profitability)**.** *When incorporating orphaned blocks in estimating the last epoch's hash rate and adjusting the difficulty accordingly in the DAM, an attacker cannot gain more block reward per time unit than mining honestly, regardless of how many epochs the attack lasts, and what strategy the attacker adopts.*

As the validity of this "incorporating uncles" mechanism does not rely on NC-Max's two-step confirmation and dynamic block interval, it can be implemented independently in NC to strengthen its security.

VII. Performance Evaluation

In this section, we experimentally evaluate the performance of NC-Max and compare it with existing designs. In particular, we first measure the block propagation latency and the orphan rate of NC and NC-Max under several transaction processing workloads and block intervals in an environment simulating the Bitcoin network. The results show that NC-Max has a stable low orphan rate independent of the throughput, therefore allows a shorter block interval with the same level of security.

Then we compare the throughput and the transaction confirmation latency of NC-Max with NC and DAG protocols. The results demonstrate that NC-Max enables the full utilization of the nodes' bandwidth, whereas all DAG protocols suffer from the duplicate packing problem. Furthermore, the transaction confirmation latency is similar to that of Prism, a recent DAG protocol featured by its high throughput and short latency.

A. Experimental Setup

Network. There are 6000 nodes, each establishes 8 random outgoing connections. The network latency between any two peers $\delta_{\text{a,b}}$ is sampled from data collected by a public Bitcoin crawler maintained by KIT [20]. All connections have the same bandwidth of 10 Mbps (1.25 MBps). The CB transmission latency δ_{CB} equals $\delta_{\text{a,b}}$. Each time a node receives a CB, the block preparation latency δ_{prep} is a function of n_{fresh}:

$$\delta_{\text{prep}} = \begin{cases} \max\{n_1, 0\}, & n_{\text{fresh}} = 0 \\ \max\{1.33233 \times 10^{-4} n_{\text{fresh}} \\ \quad + 0.544959 + n_2, \\ \delta_{\text{a,b}} + 3.6 \times 10^{-4} n_{\text{fresh}}\}, & n_{\text{fresh}} > 0 \end{cases}, \quad (2)$$

where $\delta_{\text{a,b}} + 3.6 \times 10^{-4} n_{\text{fresh}}$ is the bandwidth constraint assuming each transaction is 450 bytes; n_1 and n_2 are random variables encompassing the variation of local-message-processing and network latencies: $n_1 \sim \mathcal{N}(0.101102, 0.0431702^2)$, $n_2 \sim \mathcal{N}(0, 0.420209^2)$, whose parameters have been learned via maximum likelihood estimation from the data we collected from the Bitcoin network (Sect. III-C). Parameters n_1, n_2, $\delta_{\text{a,b}}$, and δ_{prep} are in the unit of one second. Equation (2) is chosen based on our understanding of the block propagation: when $n_{\text{fresh}} = 0$, there is no round trip to query the fresh transactions; when $n_{\text{fresh}} > 0$, δ_{prep} grows linearly with n_{fresh}. We choose not to sample from the measurement data directly because that data misses certain n_{fresh} values. This delay also applies to NC-Max for querying transactions in the proposal zone. In both NC and NC-Max, the propagation of CBs is not affected by synchronizing fresh transactions; for each node, different blocks' δ_{prep} do not overlap each other to avoid violating the bandwidth constraint.

Mining. There are altogether 20 miners whose mining power distribution follows Bitcoin's on September 15, 2019 [67]. The top five miners control 17.42%, 17.39%, 16.63%, 13.51%, and 8.65% of mining power, respectively. The PoW is replaced with a scheduler that triggers block generation events at intervals following an exponential distribution with $\overline{t_{\text{in}}}$ as an input. An orphaned block emerges if it is mined before the latest block is propagated to its miner, therefore more than one block may be orphaned at the same height.

Transaction Packing and Data Collection. We consider three transaction packing workloads: 100, 1000 and 2500 TPS, and eight average block intervals $\overline{t_{\text{in}}}$: 1 to 5, 10, 20 and 40 seconds. In the 100 TPS setting, 100 transactions are generated per second; each block packs up to $100\overline{t_{\text{in}}}$ transactions. Other settings follow similar constraints. The 2500 TPS setting exhausts the nodes' 10 Mbps bandwidth to synchronize transactions. The number of fresh transactions n_{fresh} is sampled from our second experiment in Sect. III-C with $\overline{t_{\text{in}}}$, the time since last block t_{in}, and the transaction packing workload as inputs. Note that the initial transaction propagation does not affect the block propagation as blocks have priority in propagation over transactions. A simulation is executed 40 times; each instantiates a new network topology and generates 200 blocks.

Accuracy. Our simulation assumes a uniform bandwidth and a stable network topology, which does not account for heterogeneous node resources and the joining and leaving of nodes. Yet NC's block propagation latency and its distribution in our simulation match well with the Bitcoin network, verifying the reasonableness of the environment.

B. Block Propagation Latency

Figure 6 shows the time distribution for blocks to propagate to 50% and 90% of nodes in NC and NC-Max. We only display one setting for NC-Max as all settings follow similar patterns.

In the NC, $\overline{t_{\text{in}}} = 40$, 100 TPS setting, 50% of blocks contain no fresh transaction, which finish propagation within 600 ms; the other 50% take more than 2 seconds to finish propagation. These results match well with Bitcoin's current situation: when $\overline{t_{\text{in}}} = 600$, most blocks' propagation delay is within 600 ms, with a few exceptions over 2 seconds [20]. When $\overline{t_{\text{in}}}$ decreases, fewer and fewer blocks contain no fresh transaction. Also as $\overline{t_{\text{in}}}$ decreases, blocks with fresh transactions propagate slightly faster, because the block size decreases along with $\overline{t_{\text{in}}}$ in a fixed-TPS setting. However, the speedup is not proportional to the block size, especially in the worst case, as a smaller $\overline{t_{\text{in}}}$ also means a higher fraction of fresh transactions. At last, blocks propagate slower as the throughput increases, since the bandwidth, rather than the latency, becomes the bottleneck.

In NC-Max, when w_{close} is large enough, the latency is not affected by the block interval, and only marginally affected by the block size. In the 100 TPS setting, the 50% block propagation latency concentrates at two values: 450 ms and 520 ms, because sometimes it takes a block four hops to reach the 50th percentile, sometimes it takes five. Other settings have similar results, except for abiding by the bandwidth constraint.

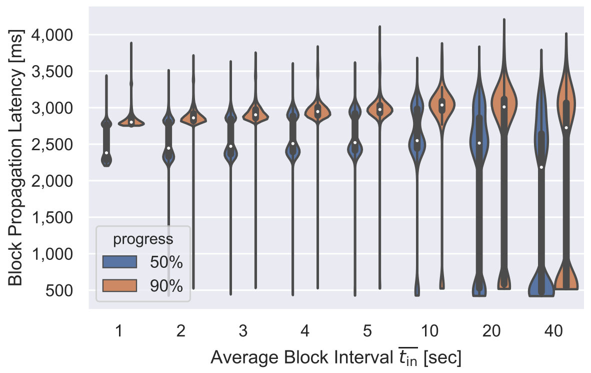
C. Orphan Rate

In line with our DAM, an orphan rate here is computed as the number of orphaned blocks, divided by the number of main chain blocks. As displayed in Fig. 7, in NC, as the throughput increases, the orphan rate deteriorates, since larger blocks take longer to propagate; whereas in NC-Max, the orphan rate is almost independent of the throughput. Consequently, NC-Max reduces the block interval with the same orphan rate. For example, when $\overline{t_{\text{in}}} = 4$, the orphan rate is 6% in NC-Max; whereas in NC, the same orphan rate demands $\overline{t_{\text{in}}} = 20$ with 100 TPS, $\overline{t_{\text{in}}} = 40$ with 1000 TPS, $\overline{t_{\text{in}}} > 40$ with 2500 TPS.

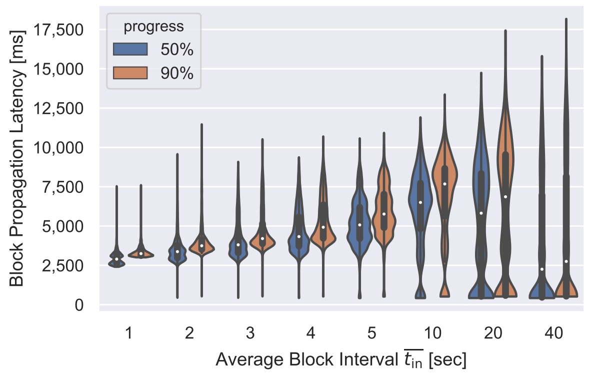
D. Minimum Propose-Commit Distance

NC-Max's near-constant block propagation latency and low orphan rate require that newly committed transactions have finished synchronization, which is ensured by the minimum propose-commit distance w_{close}. The minimum w_{close} to guarantee performance, denoted as $w^{\min}_{\text{close}}$, is a function of (1) $\overline{t_{\text{in}}}$, (2) the block size limit S, and (3) the network condition $\mathcal{Z}$, which encompasses the network's topology, the joining and leaving of nodes, and the nodes' heterogeneous resources.

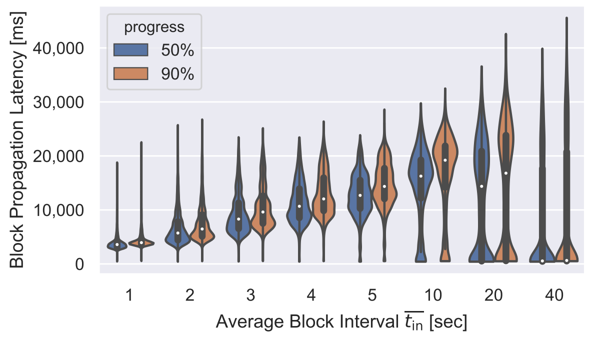
We display $w^{\min}_{\text{close}}(\overline{t_{\text{in}}}, S, \mathcal{Z})$ and how the orphan rate raises when $w_{\text{close}} < w^{\min}_{\text{close}}$ in Fig. 8. For NC-Max, 100 TPS and all other omitted settings, $w^{\min}_{\text{close}} = 2$. The orphan rate is stable



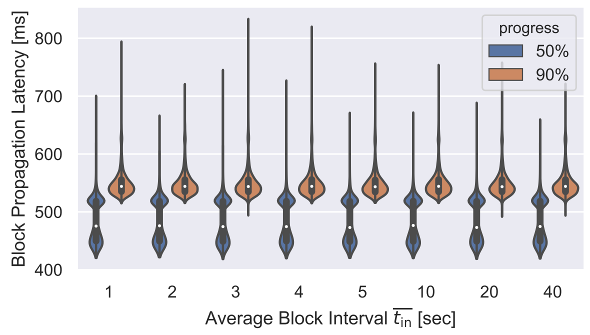
(a) NC, 100 TPS.



(b) NC, 1000 TPS.



(c) NC, 2500 TPS.



(d) NC-Max, 100 TPS. For all data in this figure, $w_{\text{close}} = 2$.

Fig. 6: Blocks propagate faster in NC-Max than in NC. Latency distribution is displayed as the kernel density estimation [71]; wider value indicates higher density. The median is displayed as a white dot.

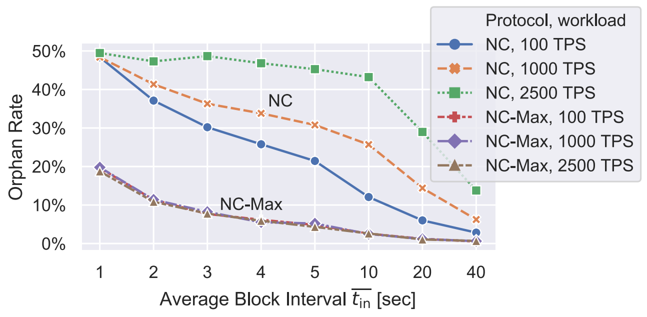


Fig. 7: Orphan rates. NC-Max with large-enough w_{close}.

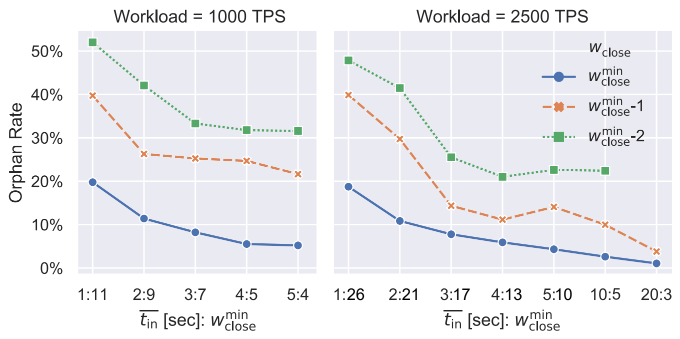


Fig. 8: The orphan rate increases when $w_{\text{close}} < w^{\text{min}}_{\text{close}}$.

when $w_{\text{close}} \geq w^{\text{min}}_{\text{close}}$; otherwise it is sensitive to w_{close} even when $\overline{t_{\text{in}}}$ is small: a change from 26 to 25 in the $\overline{t_{\text{in}}} = 1$, 2500 TPS setting doubles the orphan rate.

Due to the dynamic nature of $\mathcal{Z}$, we do not claim that our $w^{\text{min}}_{\text{close}}$ values are directly applicable in practice. We suggest two mechanisms to ensure a large-enough w_{close}. First, to hard-code some lower bounds on w_{close} based on the expected block interval of the epoch. Second, to dynamically adjust w_{close} based on the block propagation information of the last epoch, in line with our DAM.

E. Throughput

Modeling DAG Protocols. We model two types of DAG protocols: Prism [1] and the others [4], [42], [63], [64]. In Prism, each transaction has a *color*, possibly determined by its txid; all transactions in a *transaction block* are of the same color. Each miner maintains, for each color, a queue of unconfirmed transactions, and simultaneously mines on all queues. We simulate Prism with two, four, and six colors. Other DAG protocols have no prescribed transaction inclusion rules, except that blocks do not include transactions in their predecessor blocks. For simplicity, we only implement two extreme transaction inclusion strategies: the lower bound and

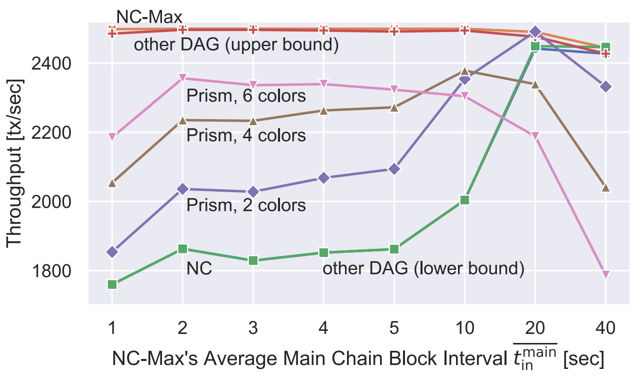


Fig. 9: NC-Max—in orange—outperforms other protocols in throughput. The lines of NC—in blue—and other DAG (lower bound)—in green—overlap except when $\overline{t_{\text{in}}^{\text{main}}} = 40$.

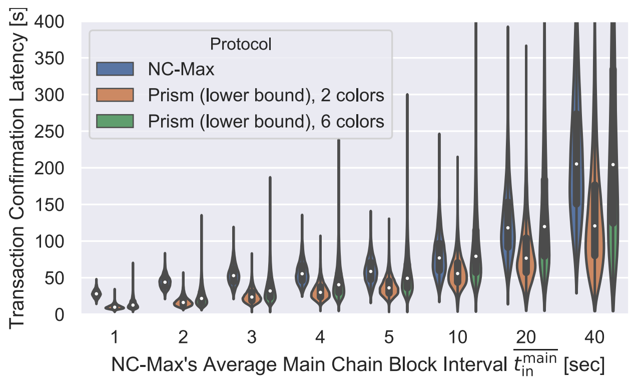


Fig. 10: NC-Max achieves similar transaction confirmation latency with Prism. The latency of Prism is simulated as lower bounds as we did not implement its block-ordering logic. Values over 400 seconds are omitted.

the upper bound. In the former setting, miners pack from the oldest transactions; in the latter, miners pack uniformly at random from their pools.

Additional Setup. To test whether NC-Max can exhaust the nodes' bandwidth, in each setting, we target a fixed main chain block interval $\overline{t_{\text{in}}^{\text{main}}}$ in NC-Max and adjust $\overline{t_{\text{in}}}$ accordingly. To ensure fairness, we apply the same sequence of block generation events to all protocols. Transactions are generated at an even speed of 2500 per second, capping the throughput as the physical limit. A block packs up to $2500\overline{t_{\text{in}}^{\text{main}}}$ transactions from the miner's pool. The pool size has a 300 MB limit—Bitcoin's peak size achieved in Dec. 2017 [9]; old transactions are dropped if it is full. In Prism, the pool size is evenly split among the colors.

Results. NC-Max reaches at least 2490 TPS when $\overline{t_{\text{in}}} < 40$, outperforming all other protocols (Fig. 9). When $\overline{t_{\text{in}}} = 40$, some transactions are lost as sometimes the transactions generated between consecutive blocks exceed the pool size limit.

Other DAG (upper bound) performs slightly worse than NC-Max, as a small fraction of transactions are included multiple times in *competing blocks*, wasting the processing capacity. Here we slightly abuse the term competing blocks to denote a group of blocks with no predecessor relation among them. Other DAG (lower bound) has almost the same throughput with NC, because when miners pack from the oldest transactions, competing blocks contain almost the same set of transactions unless the pool size is smaller than the block size—which only happens when $\overline{t_{\text{in}}^{\text{main}}} = 40$. In reality, the throughput of a DAG protocol should be between the lower and the upper bounds: to maximize their fees, miners prefer transactions with higher fees and but would randomize their selection to decrease the overlap with other miners' blocks [41].

Although Prism's throughput increases along with the number of colors, there are two caveats. First, more colors lead to longer transaction confirmation latency, as demonstrated in the next simulation. Second, the randomized nature of mining results in uneven processing speed among the queues, leading some queues to exceed their memory limits when some others are almost empty. In particular, when $\overline{t_{\text{in}}^{\text{main}}} > 10$, more colors lead to lower throughput. It is not trivial to dynamically allocate the memory among the colors without introducing new denial-of-service attack vectors.

We note that our results do not contradict DAG protocols' claims that they can exhaust the nodes' bandwidth. To do so, these protocols demand higher block frequency and larger memory pools.

F. *Transaction Confirmation Latency*

DAG protocols usually involve complicated transaction confirmation rules, therefore we only implement part of the transaction confirmation procedure of Prism to show that NC-Max achieves similar latency. We measure NC-Max's latency as $w_{\text{close}}^{\text{min}} + 6$ block intervals after the transaction is broadcast: the first block to propose it, the $(w_{\text{close}}^{\text{min}}+1)$-th block to commit it, and the last to settle it. In Prism, a transaction is confirmed in three steps: (1) a transaction block of the same color is mined, which may take several block intervals; (2) a *proposer block* whose miner has received this transaction block to propose the latter; (3) several *voter blocks* to vote for this proposer block. Further, as one block cannot be considered settled in any PoW protocol, we assume the transaction is confirmed with two voter blocks—the minimum necessary number. This simplification underestimates Prism's confirmation time, thus is in favor of Prism. We assume the same average block interval for each of the three kinds of blocks in Prism and blocks in NC-Max.

As shown in Fig. 10, two Prism instances and NC-Max achieve similar latency, especially when compared to NC, whose latency is at least four times that of NC-Max with the same orphan rate. Prism, 2 colors confirms faster than NC-Max—whose median latency is around 40% to 60% of NC-Max's, but at the price of lower throughput. Prism, 6 colors suffers from long worst-case confirmation latency—two to three times that of NC-Max, as it takes longer before a transaction meets a same-color block. Meanwhile, in general, blocks propagate faster are confirmed sooner in DAG protocols; whereas NC-Max confirms transactions at roughly even latency, providing more stable user experience.

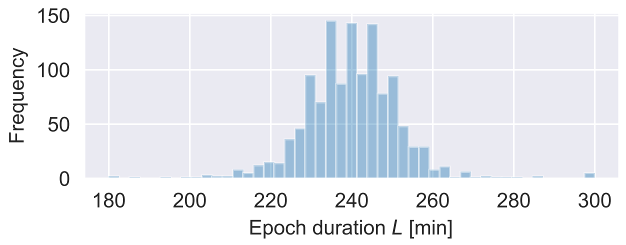


Fig. 11: In Nervos CKB, most epochs' duration is close to the four-hour target. Data collected between Nov. 16, 2019 and June 12, 2020.

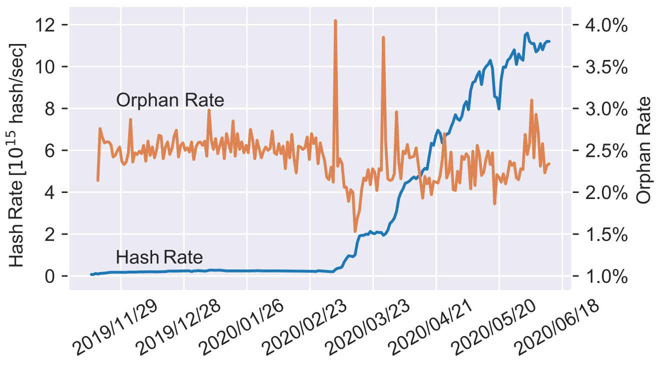


Fig. 12: The network hash rate of Nervos CKB grows 150 times in its first seven months; the orphan rate stays close to the 2.5% target. The hash rate rocketed since early March as dedicated ASIC miners enter the market. Two spikes in the orphan rate correspond to two sharp increases in the hash rate. The initial orphan rate is omitted as it is close to zero.

VIII. Implementation

NC-Max is implemented in *Nervos CKB*, a public permissionless blockchain launched in Nov. 16, 2019, and has operated smoothly since then. The PoW puzzle of Nervos CKB is a dedicated hash function called EagleSong [68]. The project's consensus and peer-to-peer communication components consist of 14K lines of code (LOC). In addition, we implement Yamux [31] and libp2p secio [56] protocols as a separate library called *Tentacle* of 2.9K LOC to handle lower-level connection details. Nervos CKB and Tentacle are implemented in Rust and are open source.

The protocol parameters are instantiated as follows. The target epoch duration $L_{\text{ideal}} = 4$ hours. Coinbase transaction outputs, i.e., block rewards, can only be spent after four epochs. The lower and upper bounds on the expected block interval are 8 and 48 seconds, respectively. Each block embeds at most two uncles, which is enough given our target orphan rate $o_{\text{ideal}} = 2.5\%$. The maximum block size is 597 KB, which translates to around 1000 two-input-two-output committed transactions and a proposal zone of at most 1500 txpids. The minimum and maximum propose-commit distances, i.e., w_{close} and w_{far}, are set to two and ten, respectively. Parameters of our DAM can be found in Appendix B.

The Nervos CKB network has grown considerably since its inception. The network hash rate, displayed in Fig. 12, has

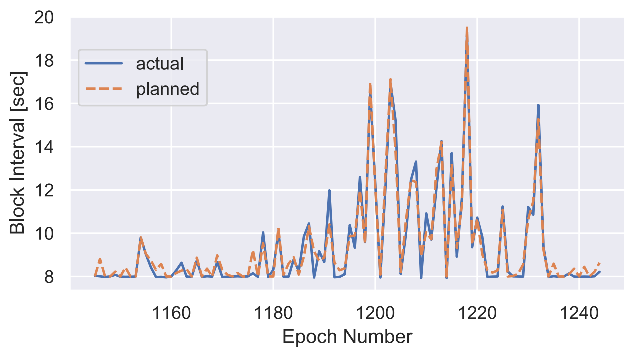


Fig. 13: The planned block interval outputted by our DAM matches well with the actual average block interval. We only display the last 100 epochs as the two lines become indistinguishable when including more epochs.

grown from 7.3×10^{13} to 1.1×10^{16} hash/sec. As of June 2020, the mining power is distributed among 12 mining pools with the largest one controlling roughly a third.

We demonstrate the effectiveness of our DAM from three aspects. First, as shown in Fig. 11, despite the constant change in the hash rate, most epochs' duration is close to L_{ideal}: 93% of epochs last between 220 and 260 minutes. Second, as shown in Fig. 13, the measured average block interval in each epoch matches well with the planned expected interval outputted by our DAM. These two aspects demonstrate the accuracy of our estimation on the total hash rate. Third, as displayed in Fig. 12, the orphan rate is stable. It goes over 3% for only three days, proving that our adjustment on the difficulty, hence the block interval, stabilizes the orphan rate. Figure 13 also reveals that the planned block interval constantly hits the eight-second lower bound, indicating that the network is capable of processing more frequent blocks.

Meanwhile, the block propagation latency is satisfactory, which can be testified by the low orphan rate: Nervos CKB achieves a 2.5% long-term average orphan rate with a 7.5-second average block interval. The block interval is shorter than the lower bound as the hash rate constantly increases. In contrast, in Ethereum, which implements a variant of NC, the long-term average orphan rate is 10% [57]; the average block interval is 13 seconds [22]. Admittedly, these two systems are not directly comparable given that Nervos CKB has a lower transaction processing workload. An in-depth analysis of the block propagation requires a well-connected network monitor, which we leave for future work.

IX. Discussion

We now further elaborate on some possible concerns.

Remaining Performance Limit. While the block size limit can increase until the throughput exhausts most nodes' bandwidth, our protocol does not support arbitrarily low block intervals as it would lead to too high an orphan rate. For example, the orphan rate may undesirably exceed 10% and discourage some miners when w_{close} is less than three seconds. Accordingly, although NC-Max reduces the transaction confirmation latency of NC, a lower bound remains on this latency. Also, the correlation between the transaction processing workload and the confirmation latency remains, as transactions propagate slower when the nodes' bandwidth is exhausted.

The major cause of orphaned blocks in NC-Max is the random nature of the mining process. Specifically, as block intervals follow an exponential distribution, it is impossible to enforce a lower bound on all of them. One approach to modify the distribution is to combine the current reversing-a-hash-function puzzle with a verifiable delay function [7].

Reasons to Embed Uncles. Another concern involves the miners' possible lack of financial incentives to embed uncles when there are not enough transactions to propose. Nevertheless, rational miners would still follow the protocol since, next to the proposal fees for embedding uncles which they are likely to receive under most situations, doing so would strengthen the system's security and, in particular, lower the possibility that the miners' blocks are orphaned in the long term. Moreover, not all honest behaviors need to be incentivized by fees. For example, Bitcoin relies on all miners not to set their clocks collectively faster to exhaust all remaining block rewards; no cryptocurrency provides any reward for nodes to forward blocks and transactions.

Pipelining vs. Concurrency. There is an emerging awareness in blockchain designs towards parallel block processing to exhaust the nodes' bandwidth. While parallel processing is mostly done through *concurrency* as in DAG protocols, we highlight the potential of *pipelining*. Compared to concurrency protocols, which often involve duplicate transaction packing [41] and complicated algorithms to order blocks [42], [63], [64], a pipelining protocol, such as NC-Max, enjoys the advantage of simplicity, which leads to stronger security and functionality. Although pipelining protocols, including the recent Byzantine fault tolerance protocol Hotstuff [73], mandate an additional latency, in NC-Max, this propose-commit distance is to ensure that all nodes have received the transactions, which is necessary for all blockchains—including DAGs—before the transactions are considered confirmed. Explicitly mandating this latency strengthens the security, as demonstrated in our analysis of transaction withholding attacks (Sect. VI-C).

In sum, as our pipelining approach already enables the full utilization of the nodes' bandwidth, we avoid introducing more complex rules—which often lead to more attack vectors and more limitations on functionality—to the consensus protocol. We believe such an approach is meaningful as it allows an easy transfer of our knowledge around NC, which we, as scholars and engineers, collectively accumulated in the last decade.

X. Related Work

We discuss some related work here in addition to Sect. II-B. We omit alternative designs that leverage more security assumptions than NC's, and refer interested readers to Bano et al. [3] for these non-PoW and hybrid consensus protocols. Neither do we include off-chain [30] and sharding [70] protocols, where not all transactions are synchronized by all nodes. These protocols are compatible with NC-Max, and therefore can further increase a system's throughput.

Parallel Blocks. The GHOST protocol designed by Sompolinsky and Zohar [65] demands that during a block race, miners select the branch with the largest number of blocks, rather than the longest chain. GHOST resists better against double-spending than NC when the block interval is short and natural forks happen frequently. However, Kiayias and Panagiotakos indicated that GHOST enables an attacker to slow down transaction confirmation via a *balancing attack* [36].

The Inclusive protocol proposed by Lewenberg et al. [41] also enables transaction confirmation by competing blocks. NC-Max inherits this key idea by allowing uncles to propose transactions while avoiding redundant transmission as each transaction is proposed with its `txpid`.

Decoupled Consensus. Bitcoin-NG by Eyal et al. [23] decouples NC's leader election and transaction serialization, allowing the nodes' full bandwidth to be utilized to confirm transactions. Unlike NC-Max, which shortens NC's transaction confirmation latency, this latency remains the same in Bitcoin-NG as in NC.

Novel DAM. Miller suggested a DAM targeting a one-to-one orphan-to-main-chain-block ratio, to minimize the block interval and to remove the block timestamp [62]. However, it is unclear how to distribute rewards at a constant speed without the timestamps. Moreover, Rosenfeld indicated that such a high orphan rate downgrades the attack difficulty [60].

The fact that selfish mining's profitability roots in the DAM was observed by Gervais et al. [27] and proved by Grunspan, and Pérez-Marco [29]. The latter study suggests incorporating uncles in the DAM to thwart the attack. Their proof on the new DAM's effectiveness does not cover the case where the attacker adapts to the modified mechanism and may temporarily turn off some mining equipment, as indicated by Negy et al. [50]. Ethereum also incorporates uncles in its DAM [11].

Block Propagation Acceleration Techniques. The FIBRE network maintained by Corallo [15] deploys several high-speed nodes across the globe to help miners synchronize blocks the moment they are found. Bloxroute by Klarman et al. [37] is a network design that deploys high-speed nodes that start to relay blocks before receiving the full content. Both approaches are centrally coordinated. Although centralized systems can further accelerate the block propagation, the P2P acceleration techniques, including the one we proposed, serve as safety nets that are immune to single points of failure.

XI. Conclusion

While the current scholarly and engineering efforts to improve blockchains' performance concentrate on designing innovative consensus protocols, we alternatively highlight the importance of the network layer in the issue. Through optimizing NC's block propagation mechanism on both the consensus and the network layer with a two-step pipelining mechanism, we break the throughput limit of NC without compromising security. With NC-Max, we call for further attention on the interaction between consensus protocols and the peer-to-peer network in protocol design.

References

[1] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 585–602.

[2] L. Bahack, "Theoretical Bitcoin attacks with less than half of the computational power (draft)," *arXiv preprint arXiv:1312.7013*, 2013, https://arxiv.org/pdf/1312.7013.pdf.

[3] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," *CoRR*, vol. abs/1711.03936, 2017, http://arxiv.org/abs/1711.03936.

[4] I. Bentov, P. Hubácek, T. Moran, and A. Nadler, "Tortoise and Hares consensus: the Meshcash framework for incentive-compatible, scalable cryptocurrencies," *IACR Cryptology ePrint Archive*, 2017, https://eprint.iacr.org/2017/300.pdf.

[5] G. Bissias and B. N. Levine, "Bobtail: A proof-of-work target that minimizes blockchain mining variance (draft)," *CoRR*, vol. abs/1709.08750, 2017, http://arxiv.org/abs/1709.08750.

[6] Blockchain, "Bitcoin block explorer," 2017, https://blockchain.info/.

[7] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science, vol. 10991. Springer, 2018, pp. 757–788.

[8] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," in *Proceedings of the 2019 ACM Conference on Economics and Computation*, ser. EC '19. ACM, 2019, pp. 459–473. [Online]. Available: http://doi.acm.org/10.1145/3328526.3329567

[9] BTC.com, "Unconfirmed transactions in Bitcoin," 2020, https://btc.com/stats/unconfirmed-tx.

[10] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014, https://github.com/ethereum/wiki/wiki/White-Paper.

[11] ——, "Change difficulty adjustment to target mean block time including uncles," 2016, https://github.com/ethereum/EIPs/blob/master/EIPS/eip-100.md.

[12] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of Bitcoin without the block reward," in *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. ACM, 2016, pp. 154–167, http://doi.acm.org/10.1145/2976749.2978408.

[13] CoinMarketCap, "Cryptocurrencies' market capitalizations: Historical snapshot," 2017, https://coinmarketcap.com/historical/20180107/.

[14] M. Corallo, "Compact block relay," 2016, https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki.

[15] ——, "Public highly optimized fibre network," 2019, http://bitcoinfibre.org/public-network.html.

[16] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security*, ser. LNCS, vol. 9604. Springer, 2016, pp. 106–125.

[17] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 23–41.

[18] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 66–98.

[19] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *13th IEEE International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2013.

[20] DNS Research Group, KASTEL @ KIT, "Bitcoin network monitor," 2019, https://dsn.tm.kit.edu/bitcoin/.

[21] T. Duong, A. Chepurnoy, and H.-S. Zhou, "Multi-mode cryptocurrency systems," in *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*. ACM, 2018, pp. 35–46.

[22] Ethstats, "Ethereum network status," 2020, https://ethstats.net/.

[23] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, 2016, pp. 45–59. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal

[24] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454.

[25] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol with chains of variable difficulty," in *Advances in Cryptology – CRYPTO 2017*, ser. LNCS, vol. 10401. Springer, 2017, pp. 291–323.

[26] J. A. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *EUROCRYPT*, 2015, pp. 281–310.

[27] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. ACM, 2016, pp. 3–16, http://doi.acm.org/10.1145/2976749.2978341.

[28] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[29] C. Grunspan and R. Pérez-Marco, "On profitability of selfish mining," *arXiv preprint arXiv:1805.08281*, 2018, https://arxiv.org/pdf/1805.08281.pdf.

[30] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Off the chain transactions," Cryptology ePrint Archive, Report 2019/360, 2019, https://eprint.iacr.org/2019/360.

[31] Hashicorp, "Yamux (yet another multiplexer)," 2020, https://github.com/hashicorp/yamux.

[32] E. Heilman, "One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner." Cryptology ePrint Archive, Report 2014/007, 2014, https://eprint.iacr.org/2014/007.

[33] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 129–144.

[34] S. Kanjalkar, J. Kuo, Y. Li, and A. Miller, "Short paper: I can't believe it's not stake! resource exhaustion attacks on PoS," in *Financial Cryptography and Data Security*, 2019.

[35] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, "Blockchain mining games," in *Proceedings of the 2016 ACM Conference on Economics and Computation*. ACM, 2016, pp. 365–382.

[36] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain," *IACR Cryptology ePrint Archive*, 2016, https://eprint.iacr.org/2016/545.pdf.

[37] U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer, "bloxroute: A scalable trustless blockchain distribution network whitepaper," *IEEE Internet of Things Journal*, 2018.

[38] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing Bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th conference on USENIX Security Symposium*, 2016.

[39] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," *Proc. 38th IEEE Symposium on Security and Privacy*, pp. 583–598, 2018.

[40] S. D. Lerner, "DECOR+HOP: A scalable blockchain protocol," 2015, https://scalingbitcoin.org/papers/DECOR-HOP.pdf.

[41] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Financial Cryptography and Data Security*, ser. LNCS, vol. 8975. Springer, 2015, pp. 528–547.

[42] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 USENIX Annual Technical Conference*, 2020, pp. 515–528.

[43] K. Liao and J. Katz, "Incentivizing blockchain forks via whale transactions," in *Financial Cryptography and Data Security*, ser. LNCS, vol. 10323. Springer, 2017, pp. 264–279.

[44] mapofcoins, "Map of coins: BTC map," 2018, http://mapofcoins.com/bitcoin.

[45] W. Martino, M. Quaintance, and S. Popejoy, "Chainweb: A proof-of-work parallel-chain architecture for massive throughput," 2018, http://kadena.io/docs/chainweb-v15.pdf.

[46] G. Maxwell, "Advances in block propagation," 2017, https://www.youtube.com/watch?v=EHIuuKCm53o.

[47] A. Miller, "Feather-forks: enforcing a blacklist with sub-50% hash power," 2013, https://bitcointalk.org/index.php?topic=312668.0.

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

[49] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE. IEEE, 2016, pp. 305–320.

[50] K. A. Negy, P. R. Rizun, and E. G. Sirer, "Selfish mining re-examined," in *Financial Cryptography and Data Security - 24th International Conference*, ser. Lecture Notes in Computer Science, vol. 12059. Springer, 2020, pp. 61–78.

[51] J. Niu and C. Feng, "Selfish mining in Ethereum," in *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019*. IEEE, 2019, pp. 1306–1316.

[52] S. Park, A. Kwon, G. Fuchsbauer, P. Gaži, J. Alwen, and K. Pietrzak, "Spacemint: A cryptocurrency based on proofs of space," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 480–499.

[53] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 643–673.

[54] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, ser. PODC '17. ACM, 2017, pp. 315–324, http://doi.acm.org/10.1145/3087801.3087809.

[55] A. Poelstra, "Scriptless scripts," 2017, https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf.

[56] Protocol Labs, "Libp2p secio: a secure transport module for go-libp2p," 2020, https://github.com/libp2p/go-libp2p-secio.

[57] YCharts Inc., "Ethereum uncle rate," 2019, https://ycharts.com/indicators/ethereum_uncle_rate.

[58] F. Ritz and A. Zugenmaier, "The impact of uncle rewards on selfish mining in Ethereum," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. IEEE, April 2018, pp. 50–57.

[59] P. R. Rizun, "Subchains: A technique to scale Bitcoin and improve the user experience," *Ledger*, 2016, https://www.ledgerjournal.org/ojs/index.php/ledger/article/view/40.

[60] M. Rosenfeld, "Re: on the optimal difficulty setting for Bitcoin," 2012, https://bitcointalk.org/index.php?topic=98314.msg1075710#msg1075710.

[61] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *Financial Cryptography and Data Security*, ser. LNCS, vol. 9603. Springer, 2016, pp. 515–532.

[62] socrates1024, "on the optimal difficulty setting for Bitcoin," 2012, https://bitcointalk.org/index.php?topic=98314.msg1075701#msg1075701.

[63] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: Serialization of proof-of-work events: Confirming transactions via recursive elections," 2016, https://eprint.iacr.org/2016/1159.pdf.

[64] Y. Sompolinsky, S. Wyborski, and A. Zohar, "PHANTOM and GHOSTDAG: A scalable generalization of Nakamoto Consensus," *IACR Cryptology ePrint Archive*, 2020, https://eprint.iacr.org/2018/104.pdf.

[65] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in Bitcoin," in *Financial Cryptography and Data Security*, ser. LNCS, vol. 8975. Springer, 2015, pp. 507–527.

[66] ——, "Bitcoin's security model revisited," *arXiv preprint arXiv:1605.09193*, 2016, https://arxiv.org/pdf/1605.09193.pdf.

[67] M. P. Stats., "Bitcoin mining pools," 2019.

[68] A. Szepieniec and T. Ashur, "Eaglesong: an arx hash with fast diffusion," *Proceedings of the Romanian Academy Series A—Mathematics Physics Technical Sciences Information Science*, vol. 21, no. 1, pp. 69–76, 2020.

[69] I. Tsabary and I. Eyal, "The gap game," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. ACM, 2018, pp. 713–728. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243737

[70] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, ser. AFT 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 4161. [Online]. Available: https://doi.org/10.1145/3318041.3355457

[71] M. Waskom, "Kernel density estimation," 2019, https://seaborn.pydata.org/tutorial/distributions.html#distribution-tutorial.

[72] P. Wei, Q. Yuan, and Y. Zheng, "Security of the blockchain against long delay attack," in *Advances in Cryptology–ASIACRYPT 2018*, ser. LNCS, vol. 11274. Springer, 2018.

[73] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[74] R. Zhang and B. Preneel, "On the necessity of a prescribed block validity consensus: Analyzing Bitcoin Unlimited mining protocol," in *13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, Dec. 2017, pp. 108–119.

[75] ——, "Publish or Perish: A backward-compatible defense against selfish mining in Bitcoin," in *The Cryptographers' Track at the RSA Conference (CT-RSA)*, ser. LNCS, vol. 10159. Springer, Feb. 2017, pp. 277–292.

[76] ——, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *40th IEEE Symposium on Security and Privacy (S&P)*. IEEE, May 2019, pp. 1190–1207.

Appendix A
Parameter Examples

Here is a typical set of parameters that satisfy the requirements mentioned in Sect. IV-B, assuming the block size limit is 1 MB. A header is 176 bytes, including 80 bytes of NC block header and three 32-byte Merkle roots. A proposal zone is 41 403 bytes at most, including 2300 18-byte txpids and a three-byte transaction count. For a block with one uncle, the header, the proposal zone, and the uncle header consume 41 755 bytes, which leaves 1 006 821 bytes in a block for the commitment zone, that can accommodate 2237 transactions of 450 bytes each. The corresponding CB includes 41 755 bytes for two block headers and the proposal zone, 13 422 bytes for shortids, 450 bytes for the coinbase transaction and less than 20 bytes of other metadata, in total less than 55 KB. The size becomes 95 KB if the uncle's proposal zone is included. Whether or not to include uncles' proposal zones in CBs depends on whether they are already synchronized in this connection. There is no need to propose transactions that are already proposed in the uncles, which reduces the size of CBs.

Appendix B
Our Difficulty Adjustmnet Mechanism

A. Inputs and Outputs

Similar to NC, NC-Max's DAM is executed at the end of every epoch. It takes four inputs:

T_i	last epoch's target
L_i	last epoch's duration—the timestamp difference between epoch i and $i-1$'s last blocks
$C_{i,\mathrm{m}}$	last epoch's main chain block count
$C_{i,\mathrm{o}}$	last epoch's orphaned block count—the number of uncles embedded in epoch i's main chain

Among these inputs, T_i and $C_{i,\mathrm{m}}$ are decided by the last DAM iteration; L_i and $C_{i,\mathrm{o}}$ are measured after the epoch ends. The orphan rate o_i is calculated as $C_{i,\mathrm{o}}/C_{i,\mathrm{m}}$. We do not include $C_{i,\mathrm{o}}$ in its denominator to simplify the equations. As some orphans at the end of the epoch might be excluded from the main chain by an attack, o_i is a lower bound of the actual number. However, the proportion of deliberately excluded orphans is negligible as long as the epoch is long enough, as the difficulty of orphaning a chain grows exponentially with the chain length [26]. The algorithm outputs three values:

T_{i+1}	next epoch's target
$C_{i+1,\mathrm{m}}$	next epoch's main chain block count
r_{i+1}	next epoch's block reward

If the network hash rate and block propagation latency remain constant, o_{i+1} should reach the ideal value o_{ideal}, unless $C_{i+1,\mathrm{m}}$ equals its upper bound $C_{\mathrm{m}}^{\max}$ or its lower bound $C_{\mathrm{m}}^{\min}$. Epoch $i+1$ ends when it reaches $C_{i+1,\mathrm{m}}$ main chain blocks regardless of how many uncles are embedded.

B. Estimating Last Epoch's Hash Rate and Block Propagation Parameters

Adjusted Hash Rate Estimation. The adjusted hash rate estimation, denoted as $\hat{H}_i$, is computed by applying a dampening factor τ_1 to last epoch's *actual hash rate* $\hat{H}_i'$. The actual hash rate is calculated as follows:

$$\hat{H}_i' = \frac{\mathrm{HSpace}}{T_i} \cdot (C_{i,\mathrm{m}} + C_{i,\mathrm{o}})/L_i \ , \tag{3}$$

where HSpace is the size of the entire hash space, e.g., 2^{256} in Bitcoin, HSpace/T_i is the expected number of hash operations to find a valid block, and $C_{i,\mathrm{m}} + C_{i,\mathrm{o}}$ is the total number of blocks in epoch i. $\hat{H}_i'$ is computed by dividing the expected total number of hash operations by the duration L_i.

Now we apply the dampening filter:

$$\hat{H}_i = \begin{cases} \hat{H}_{i-1} \cdot \frac{1}{\tau_1}, & \hat{H}_i' < \hat{H}_{i-1} \cdot \frac{1}{\tau_1} \\ \hat{H}_{i-1} \cdot \tau_1, & \hat{H}_i' > \hat{H}_{i-1} \cdot \tau_1 \\ \hat{H}_i', & \text{otherwise} \end{cases} ,$$

where $\hat{H}_{i-1}$ denotes the adjusted hash rate estimation outputted by the last DAM iteration. The dampening factor ensures that the adjusted hash rate estimation does not change by more than a factor of τ_1—instantiated as 2 in Nervos CKB—between two consecutive epochs so that our DAM satisfies **C3**.

Modeling the Block Propagation. It is difficult, if not impossible, to model the detailed block propagation procedure, given that the network topology is unknown. Luckily, for our purpose, it suffices to describe the block propagation with two parameters, which will be used to compute $C_{i+1,\mathrm{m}}$ later. We learn some information on these parameters by computing the expected hash operations working on orphaned blocks in two different ways and then connect the expressions.

We assume all blocks follow a similar propagation model, in line with [16], [19]. In the last epoch, it takes d seconds for a block to propagate to the entire network, and during this process, the average fraction of mining power working on the block's parent is p. Therefore, during these d seconds, $\hat{H}_i' \times dp$ hash operations work on the latest block's parent, thus do not contribute to extending the blockchain. Consequently, in the last epoch, the total number of hashes that do not extend the blockchain is $\hat{H}_i' \times dp \times C_{i,\mathrm{m}}$.

On the other hand, the number of hash operations working on observed orphaned blocks is $\mathrm{HSpace}/T_i \times C_{i,\mathrm{o}}$. When the orphan rate is relatively low, we can ignore the rare event that more than two competing blocks are found at the same height. Therefore we have

$$\hat{H}_i' \times dp \times C_{i,\mathrm{m}} = \mathrm{HSpace}/T_i \times C_{i,\mathrm{o}} \ . \tag{4}$$

If we combine Eqn. (3) and (4), we can solve dp:

$$dp = \frac{\mathrm{HSpace}/T_i \times C_{i,\mathrm{o}}}{\hat{H}_i' \times C_{i,\mathrm{m}}} = \frac{o_i \times L_i}{(1+o_i)C_{i,\mathrm{m}}} \ , \tag{5}$$

C. Computing the Outputs

Main Chain Block Count. If the next epoch's block propagation situation is identical to the last epoch's, the value dp should remain unchanged. In order to achieve the ideal orphan rate o_{ideal} and **C1**—the ideal epoch duration L_{ideal}, following the same reasoning with Eqn. (5), we should have

$$dp = \frac{o_{\mathrm{ideal}} \times L_{\mathrm{ideal}}}{(1+o_{\mathrm{ideal}})C_{i+1,\mathrm{m}}'} \ , \tag{6}$$

where $C_{i+1,\mathrm{m}}'$ is the number of main chain blocks in the next epoch, if our only goal is to achieve o_{ideal} and L_{ideal}.

By combining Eqn. (5) and (6), when $o_i \neq 0$, we can solve $C_{i+1,\mathrm{m}}'$:

$$C_{i+1,\mathrm{m}}' = \frac{o_{\mathrm{ideal}}(1+o_i) \times L_{\mathrm{ideal}} \times C_{i,\mathrm{m}}}{o_i(1+o_{\mathrm{ideal}}) \times L_i} \ . \tag{7}$$

Now in order to achieve **C4**, we can apply the upper and lower bounds to $C_{i+1,\mathrm{m}}'$ and get $C_{i+1,\mathrm{m}}$:

$$C_{i+1,\mathrm{m}} = \begin{cases} \min\{C_{\mathrm{m}}^{\max}, \tau_2 C_{i,\mathrm{m}}\}, & \\ \quad o_i = 0 \text{ or } C_{i+1,\mathrm{m}}' > \min\{C_{\mathrm{m}}^{\max}, \tau_2 C_{i,\mathrm{m}}\} & \\ \max\{C_{\mathrm{m}}^{\min}, C_{i,\mathrm{m}}/\tau_2\}, & \\ \qquad C_{i+1,\mathrm{m}}' < \max\{C_{\mathrm{m}}^{\max}, C_{i,\mathrm{m}}/\tau_2\} & \\ C_{i+1,\mathrm{m}}', & \text{otherwise} \end{cases} . \tag{8}$$

Equation (8) also covers the case of $o_i = 0$. When L_{ideal} is fixed (**C1**), a lower bound on $C_{i+1,\mathrm{m}}$ is equivalent to an upper bound on the expected block interval $\overline{t_{\mathrm{in}}}$, and vice versa. The dampening factor τ_2, also instantiated as 2 in Nervos CKB, serves both **C3** and **C4**, preventing the main chain block number from changing too fast.

Target. To compute the target, we introduce an adjusted orphan rate estimation o_{i+1}':

$$o_{i+1}' = \begin{cases} 0, & o_i = 0 \\ o_{\mathrm{ideal}}, & C_{i+1,\mathrm{m}} = C_{i+1,\mathrm{m}}' \\ 1/(\frac{(1+o_i)L_{\mathrm{ideal}}C_{i,\mathrm{m}}}{o_i L_i C_{i+1,\mathrm{m}}} - 1), & \text{otherwise} \end{cases} .$$

Using o'_{i+1} instead of o_{ideal} prevents some undesirable situations when $C_{i+1,\text{m}}$ reaches its upper or lower bound. Now we can compute T_{i+1}:

$$T_{i+1} = \text{HSpace} / \frac{\hat{H}_i \cdot L_{\text{ideal}}}{(1 + o'_{i+1}) \cdot C_{i+1,\text{m}}} \quad , \tag{9}$$

where $\hat{H}_i \cdot L_{\text{ideal}}$ is the total number of hashes, $(1 + o'_{i+1}) \cdot C_{i+1,\text{m}}$ is the total number of blocks. The denominator in Eqn. (9) is the number of hashes required to find a block.

Note that if none of the edge cases is triggered, i.e., $\hat{H}_i = \hat{H}'_i$ and $C_{i+1,\text{m}} = C'_{i+1,\text{m}}$, we can combine Eqn. (3), (7), (9) and get $T'_{i+1} = T_i \times o_{\text{ideal}}/o_i$. This result is consistent with our intuition. On the one hand, if o_i is larger than the ideal value o_{ideal}, the target lowers, increasing the difficulty of finding a block and raising the block interval if the hash rate is unchanged. Therefore, the orphan rate is lowered as it is more unlikely to find a block during another block's propagation. On the other hand, the target increases if the last epoch's orphan rate is lower than the ideal value, decreasing the block interval and raising the system's throughput.

Block Reward. Now we can compute the block reward:

$$r_{i+1} = \min \left\{ \frac{R(i+1)}{C_{i+1,\text{m}}}, \frac{R(i+1)}{C'_{i+1,\text{m}}} \cdot \frac{T'_{i+1}}{T_{i+1}} \right\} \quad , \tag{10}$$

where the two cases differ only in the edge cases. The first case guarantees that the total reward issued in epoch $i+1$ will not exceed $R(i+1)$, thus ensuring **C2**. The second case ensures that an attacker that maliciously triggers some edge cases to make $T_{i+1} > T'_{i+1}$ cannot get more reward with the same mining power. When $o_i = 0$, only the first case applies.

In our security proof in Sect. VI-D, we use $r_{i+1} = \text{R}/C_{i+1,\text{m}}$ and assume none of the edge cases is triggered. This is reasonable as: (1) when the mining power is stable, the dampening mechanism cannot be triggered without network-layer attacks; (2) causing $C_{i+1,\text{m}} \neq C'_{i+1,\text{m}}$ is not rational as it only lowers the attacker's block reward according to Eqn. (10).

Appendix C
Transaction Withholding Attack Simulations

Our simulation supports attackers with arbitrary mining power share and arbitrary success rate of winning a tie. To simplify the demonstration, we assume a strong attacker with $\alpha = 0.4$, who also wins all the block races by immediately pushing his blocks to all the nodes after the competing blocks are mined. We omit natural orphans and assume that honest blocks are propagated immediately after they are mined, but not before the competing attacker block if there is one. The block interval follows an exponential distribution with expectation $\overline{t_{\text{in}}}$ as an input.

In both protocols, every "delayable" attacker block is delayed up to t_{delay}, during which the first honest block, if there is one, is orphaned. The attacker only delays the last block if he mines several blocks in a row. In NC, all attacker blocks are delayable. In NC-Max, the attacker delays his latest block only if there is another attacker block in the proposal window. The NC-Max simulation has an additional parameter $w = w_{\text{far}} - w_{\text{close}} + 1$. For every set of parameters, our simulation generates a chain of two million blocks and compute ρ, the proportion of main chain blocks mined by the attacker.

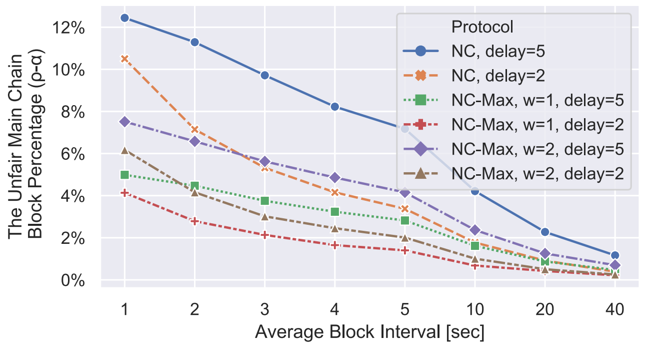


Fig. 14: NC-Max resists better against transaction withholding attacks than NC. The proposal window size $w = w_{\text{far}} - w_{\text{close}} + 1$; "delay" denotes t_{delay}, the maximum time an attacker can delay the propagation of its blocks.

The results are displayed in Fig. 14. With the same t_{delay}, when $w = 1$ and 2, NC-Max reduces the damage $\rho - \alpha$ by roughly a half and a third, respectively. Although we compute the results where $t_{\text{delay}} = 2$ and 5 seconds for both protocols, in reality, t_{delay} is shorter in NC-Max than in NC, thanks to **R2** and **R3** introduced in Sect. IV-C.

Appendix D
Proof of Theorem 1

Model. The attack lasts n epochs, with epoch number $1, 2, \cdots, n$. The total block reward per epoch is constant throughout the attack, denoted as R. We add an epoch 0 before the attack launches and assume the attacker mines honestly with full mining power α in this epoch. Note that this is just to ensure that the difficulty is well-defined throughout the attack process. Whether or not an attack strategy generates continuous unfair profit is not affected by the initial difficulty. We assume all orphans are caused by the attacker during the attack, namely, there is no naturally orphaned block. All epochs are long enough so that we can neglect deliberately excluded orphans and assume all orphans are included as uncles [26].

Notation. The time duration of epoch i is denoted as t_i. The compliant miners' mining power share is denoted as $c = 1 - \alpha$, and we immediately have $c > 0.5$. Within each epoch, the attacker distributes the mining power among honest mining, selfish mining and idle. The attacker's honest mining power share in epoch i is denoted as b_i, which extends the main chain without orphaning any honest blocks. The selfish mining power share is denoted as a_i, which extends the main chain and orphans the same number of honest blocks. The remaining mining power share $\alpha - a_i - b_i$ is temporarily idle, hoping to lower the hash rate estimation $\hat{H}'_i$. By these definitions we have $b_0 = \alpha$, $a_0 = 0$, $a_i, b_i \geq 0$ and $a_i + b_i \leq 1 - c$. For the compliant miners, $c - a_i$ mining power share extends the main chain and the rest a_i mines orphaned blocks.

The fraction of reward in epoch i that goes to the attacker is denoted as f_i. If mining honestly, f_i equals the mining power share α, and all epochs have the same duration t_0, therefore the attacker's time-averaged reward is $\alpha \times \mathrm{R}/t_0$, where $\alpha \times \mathrm{R}$ is the attacker's total reward in an epoch. We normalize both R and t_0 to 1 for simplicity so that $\alpha \times \mathrm{R}/t_0 = \alpha = 1 - c$. Similarly, the attacker's time-averaged reward in an attack is $\sum_{i=1}^{n} f_i / \sum_{i=1}^{n} t_i$, where f_i satisfies

$$f_i = \frac{a_i + b_i}{c + b_i} \ , \tag{11}$$

where $c + b_i$ is the total mining power share in extending the main chain, $a_i + b_i$ is the attacker's share. According to our DAM, we have

$$t_i = \frac{c + a_{i-1} + b_{i-1}}{c + b_i} \ , \tag{12}$$

where $c + a_{i-1} + b_{i-1}$ is the total mining power in the last epoch, estimated by our DAM by counting both main chain blocks and uncles. When $c + a_{i-1} + b_{i-1} > c + b_i$, $t_i > 1$, the main chain grows slower than epoch 0, and vice versa.

Proof. We need to prove that, regardless of how the attacker chooses n, a_i and b_i for $1 \leq i \leq n$,

$$\frac{\sum_{i=1}^{n} f_i}{\sum_{i=1}^{n} t_i} \leq 1 - c \ ,$$

where the left is the time-averaged block reward with the attack, the right is that of honest mining. Let $\sigma_k = \sum_{i=1}^{k} f_i - (1-c)\sum_{i=1}^{k} t_i$, proving the inequality is equivalent to proving $\sigma_n \leq 0$. Let $d_i = a_i + b_i$, we will prove a stronger result of $\sigma_n \leq c + d_n - 1$ by induction. First,

$$\begin{aligned} \sigma_1 = f_1 - (1-c)t_1 &= \frac{d_1 - (1-c)(c + d_0)}{c + b_1} \\ &= \frac{c + d_1 - 1}{c + b_1} \leq c + d_1 - 1 \ . \end{aligned}$$

Next, assuming for $k < n$, $\sigma_k \leq c + d_k - 1$ holds. When $k = n$,

$$\begin{aligned} \sigma_n &= \sigma_{n-1} + f_n - (1-c)t_n \\ &\leq c + d_{n-1} - 1 + \frac{d_n}{c + b_n} - (1-c)\frac{c + d_{n-1}}{c + b_n} \\ &= c + d_{n-1} - 1 + \frac{d_n - (1-c)(c + d_{n-1})}{c + b_n} \ , \end{aligned}$$

now we only need to prove

$$\begin{aligned} & c + d_{n-1} - 1 + \frac{d_n - (1-c)(c + d_{n-1})}{c + b_n} \leq c + d_n - 1 \\ \Leftrightarrow\ & \frac{(c + b_n)(d_{n-1} - d_n) + d_n - (1-c)(c + d_{n-1})}{c + b_n} \leq 0 \\ \Leftrightarrow\ & (c + b_n - 1)(d_{n-1} - d_n) + c(c + d_{n-1} - 1) \leq 0 \ . \end{aligned}$$

When $d_{n-1} \geq d_n$, the last inequality holds. Now we need to prove that it holds for $d_{n-1} < d_n$. Let $\epsilon_n = d_n - d_{n-1} > 0$, the last inequality becomes

$$\begin{aligned} & -\epsilon_n(c + b_n - 1) + c(c + d_n - \epsilon_n - 1) \leq 0 \\ \Leftrightarrow\ & c(c + d_n - 1) - \epsilon_n(2c + b_n - 1) \leq 0 \, . \end{aligned}$$