On Scaling Decentralized Blockchains

(A Position Paper)

Kyle Croman[0,1], Christian Decker[4], Ittay Eyal[0,1], Adem Efe Gencer[0,1], Ari Juels[0,2], Ahmed Kosba[0,3], Andrew Miller[0,3], Prateek Saxena[6], Elaine Shi[0,1], Emin Gün Sirer[0,1], Dawn Song[0,5], and Roger Wattenhofer[4]

[0] Initiative for CryptoCurrencies and Contracts (IC3)
[1] Cornell [2] Jacobs, Cornell Tech [3] UMD [4] ETH [5] Berkeley [6] NUS

Abstract. The increasing popularity of blockchain-based cryptocurrencies has made scalability a primary and urgent concern. We analyze how fundamental and circumstantial bottlenecks in Bitcoin limit the ability of its current peer-to-peer overlay network to support substantially higher throughputs and lower latencies. Our results suggest that reparameterization of block size and intervals should be viewed only as a first increment toward achieving next-generation, high-load blockchain protocols, and major advances will additionally require a basic rethinking of technical approaches. We offer a structured perspective on the design space for such approaches. Within this perspective, we enumerate and briefly discuss a number of recently proposed protocol ideas and offer several new ideas and open challenges.

1 Introduction

Increasing adoption of cryptocurrencies has raised concerns about their ability to scale. Since Bitcoin is a self-regulating system that works by discovering blocks at approximate intervals, its highest transaction throughput is effectively capped at maximum block size divided by block interval. The current trend of ever increasing block sizes on Bitcoin portends a potential problem where the system will reach its maximum capacity to clear transactions, probably by 2017 [46]. As a result, the cryptocurrency community has been discussing techniques for improving scalability of blockchains in general, and Bitcoin in particular, for some time. These debates have been vigorous, and at times acrimonous, and led to splits within the community, without a clear path forward on which technical measures ought to be deployed to address the scalability problem.

Today's representative blockchain such as Bitcoin takes **10 min** or longer to confirm transactions, achieves **7 transactions/sec** maximum throughput. In comparison, a mainstream payment processor such as Visa credit card confirms a transaction within seconds, and processes 2000 transactions/sec on average, with a peak rate of 56,000 transactions/sec [10]. Clearly, a large gap exists between where Bitcoin is today, and the scalability of a mainstream payment processor. Therefore, the key questions are,

Can decentralized blockchains be scaled up to match the performance of a mainstream payment processor? What does it take to get there?

This paper aims to place exploration of blockchain scalability on a scientific footing. We note that “scalability” is not a well-defined, singular property of a system, but a term that relates several quantitative metrics to each other.

We offer three contributions that illuminate the problem of scaling Bitcoin and blockchains generally to achieve high-performance, decentralized systems:

Measurement study and exploration of reparametrization. We present experimental measurements of a range of metrics that characterize the resource costs and performance of today's operational Bitcoin network. As a first step toward better scalability in Bitcoin, the community has put forth various proposals to modify the key system parameters of block size and block interval. Through further experimental investigation, we show that such scaling by reparametrization can achieve only limited benefits given the network performance induced by Bitcoin's current peer-to-peer overlay network protocol while maintaining its current degree of decentralization, as measured by number of functioning peers in the overlay network.

Our results hinge on the key metric of *effective throughput* in the overlay network, which we define here as which blocks propagate within an average block interval period the percentage of nodes to. If the transaction rate exceeds the 90% effective throughput, then 10% of the nodes in the network would be unable to keep up, potentially resulting in denied services to users and reducing the network's effective mining power. To ensure at least 90% of the nodes in the current overlay network have sufficient throughput, we offer the following two guidelines:

– [**Throughput limit.**] The block size should not exceed 4MB, given today's 10 min. average block interval (or a reduction in block-interval time). A 4MB block size corresponds to a maximum throughput of at most 27 transactions/sec.

– [**Latency limit.**] The block interval should not be smaller than 12s, if full utilization of the network's bandwidth is to be achieved.

We stress that the above guidelines seem somewhat intuitive (especially in hindsight). The community has thus also proposed radically different scaling approaches, and introduced mechanisms such as Corallo's relay network, a centralized block-propagation mechanism. One of our contributions, however, is to *quantify* Bitcoin's current scalability limits within its decentralized components. Note that as we consider only a subset of possible metrics (due to difficulty in accurately measuring others), our results on reparametrization may be viewed as upper bounds: additional metrics could reveal even stricter limits.

Painting a broad design space for scalable blockchains. Our findings lead us to the position that *fundamental protocol redesign is needed for blockchains to scale significantly while retaining their decentralization.* We compile and review various technical approaches that can help blockchains scale. We lay out a broad design space that encompasses not just incremental improvements, but also radical rearchitecture. We frame a structured discussion of new protocol design strategies in terms of a partitioning blockchain systems into distinct planes, namely: Network, Consensus, Storage, View, and Side Planes. We discuss the properties of each plane and both recent and new proposals to improve each; we also discuss open research challenges.

Posing open challenges. Another goal of our paper is to articulate open challenges in the service of *i)* better understanding of scalability bottlenecks; and *ii)* the design of more scalable blockchains. As mentioned earlier, scalability is not a singular metric, but captures the tension between various performance and security metrics. So far, measurement and understanding of many important metrics (e.g., fairness and mining power utilization [25]) are lacking — partly because monitoring and measuring a decentralized blockchain from only a few vantage points poses significant challenges. We call for better measurement techniques such that we could continuously monitor the health of decentralized system such as Bitcoin, and answer key questions such as "*To what extent can we push system parameters without sacrificing security?*" "*How robust is the system when under attack?*" Finally, although we paint the broader design space for a scalable blockchain, instantiating and combining these ideas to build a full-fledged system with formally provable security is a non-trivial challenge.

2 Bitcoin Scalability Today: A Reality Check

We analyze some of the key metrics of the Bitcoin system as it exists today.

Maximum throughput. The maximum throughput is the maximum rate at which the blockchain can confirm transactions. Today, Bitcoin's maximum throughput is 3.3–7 transactions/sec [1]. This number is constrained by the maximum block size and the inter-block time.

Latency. Time for a transaction to confirm. A transaction is considered confirmed when it is included in a block, roughly 10 minutes in expectation.[1]

Bootstrap time. The time it takes a new node to download and process the history necessary to validate the current system state. Presently in Bitcoin, the bootstrap time is linear in the size of the blockchain history, and is roughly **four days** (averaged over five fresh `t2.medium` Amazon EC2 nodes that we connected to the network running the most recent `master` software).

Cost per Confirmed Transaction (CPCT). The cost in USD of resources consumed by the entire Bitcoin system to confirm a single transaction. The CPCT encompasses several distinct resources, all of which can be further decomposed into operational costs (mainly electricity) and capital equipment costs:

1. *Mining:* Expended by miners in generating the proof of work for each block.
2. *Transaction validation:* The cost of computation necessary to validate that a transaction can spend the outputs referenced by its inputs, dominated by cryptographic verifications.

[1] Although we define latency in Bitcoin as the time to obtain a single confirmation, some payment processors accept "zero-confirmation" transactions, while others follow common advice to wait for 6 confirmations before accepting a payment.

3. *Bandwidth:* The cost of network resources required to receive and transmit transactions, blocks, and metadata.
4. *Storage:* The cost (1) of storing all currently spendable transactions, which is necessary for miners and full nodes to perform transaction validation, and (2) of storing the blockchain's (much larger) historical data, which is necessary to bootstrap new nodes that join the network.

Table 1 presents our estimates of these various costs. As the table shows, the majority of the cost is attributable to mining. Our calculation suggests that, at the maximum throughput, the cost per confirmed transaction is **\$1.4 − \$2.9**, where 57% is electricity consumption for mining. If the de facto Bitcoin throughput is assumed, the CPCT is as high as **\$6.2**. We proceed to explain our cost-estimation methodology.

To measure the cost per transaction for Bitcoin, we perform a back-of-the-envelope calculation by summing up the electricity consumed by the network as a whole, as well as the hardware cost of mining equipment. We project our estimates based on the AntMiner S5+ mining hardware [8], which is the currently available hardware that has the highest hash rate per joule, and the highest hash rate per dollar according to this comparison as of October 2015 [2]. We assume a 1 year effective lifetime for the hardware, and that the average hashing rate of the network is 450,000,000 GH/s based on statistics from October 2015 [3]. Based on the power consumption of the selected hardware (0.445 W/GH), the total power consumed by the network will be about 200 MegaWatt. Furthermore, we assume the average price per KWh is \$0.1 [48].

There are two interesting scenarios: The first scenario is when the Bitcoin network is operating at maximum throughput, namely 3.3–7 transactions/sec. This maximum throughput is mainly constrained by Bitcoin's 1MB maximum block size and the variable transaction size. The lower bound of the maximum throughput is inferred from the current average transaction size, about 500 bytes, while the upper bound is based on an oft-cited estimate from [1] which corresponds to unusually small (250 byte) transactions. The second scenario is the de facto average throughput for the Bitcoin network, which is, based on statistics collected in October 2015, 1.57 transactions/sec [4].

Table 1 shows ballpark estimates for the transaction validation, storage, and bandwidth costs. These estimates are attained assuming that the entire network contains 5400 full nodes — a coarse-grained estimate obtained from `https://bitnodes.21.co/`. We assume that each full node incurs roughly the running cost of an EC2 micro-instance ($\sim$ \$0.01/hour) to validate transactions; alternatively, assuming a \$500 processor with a 5-year life-time would yield the same ballpark estimate. We assume that transactions are stored on an SSD drive with a 5-year lifetime, costing about \$0.3/GB today. We also assume all nodes store the entire history, to maintain the system's security. We assume each node maintains a home-grade Internet connection (about \$100/month) whose cost is amortized over all transactions. We stress that an EC2 micro instance and a home-grade Internet connection provide sufficiently large computation/network bandwidth at the operating scale of today's Bitcoin.

	at max throughput		at *de facto* throughput	
	cost/tx	percentage	cost/tx	percentage
Mining: proof-of-work	$\sim \$0.8 - \$1.7$	$\sim 56\%$	$\sim \$3.6$	$\sim 56\%$
Mining: hardware	$\sim \$0.6 - \$1.3$	$\sim 42\%$	$\sim \$2.7$	$\sim 42\%$
Transaction validation	$\sim \$0.002$	$\sim 0.2\%$	$\sim \$0.008$	$\sim 0.2\%$
Bandwidth	$\sim \$0.02$	$\sim 2\%$	$\sim \$0.08$	$\sim 2\%$
Storage (running cost)	$\sim \$0.0008/5$years			

Table 1: **Bitcoin cost breakdown.** Includes cost incurred by all nodes.

We note that it is a fallacy to assume that transaction costs necessarily have to be offset by transaction fees. In particular, the operational costs of running full nodes may be offset by financial externalities, such as being able to confirm one's own transactions without trusting third parties, or by network effects, such as selling items whose costs factor in the cost of operating a node. Miners, however, are bereft of these two factors and need to be compensated in the steady state, especially as the block subsidy is reduced over time.

Other metrics. The above is of course not an exhaustive list of potentially interesting metrics. For example, while we have focused on Bitcoin's role as a transaction medium, it also serves as a store of value. We might therefore consider the *cost per stored dollar* as an alternative to CPCT. Many other metrics are of interest and it is an open research question which can best inform technical and policy decisions.

3 Scaling by Parameter Tuning and Fundamental Limits

The Bitcoin community has several propositions under discussion for increasing the maximum block size (or to remove the limit altogether [42]). Bitcoin Improvement Proposals (BIPs) 100, 101, 102, and 103, all involve a fork, triggered by a combination of time and miner buy-in as reflected in the blockchain [13, 23, 26, 27, 50]. These proposals differ primarily on the initial date of the first increase, the block size change strategy (none vs. linear vs repeated doublings vs optional reductions), and the percent of miner buy-in to trigger a change. The segregated witness proposal [23] amends the blocksize by less than a factor of 2 through a "soft fork" implementation, wherein legacy nodes do not need to upgrade, but end up implicitly trusting the miners with transaction validation. The developer community has been further fragmented into various proposals (Core, XT, Classic and Unlimited) that embody different combinations of features and rollout schedules. There is, as of yet, no clear winner, partly because it is difficult to determine, a priori, which change schedule will best fit future changes in node provisioning. It is an open question whether reparametrization alone can adequately address the growth needs of a medium-to-large transaction processing system. In the rest of this section, we explore its limitations.

3.1 Measurement Study

A critical reference point for Bitcoin's performance is Decker and Wattenhofer's 2012 measurement study of the Bitcoin network's block propagation [20]. At the time, the median and 90-percentile time for Bitcoin nodes to receive a block was 6.5 seconds and 26 seconds respectively. This study also showed that for small blocks, less than roughly 20KB in size, latency was a significant factor in block propagation times. Beyond this size, throughput was the dominating factor, and was invariant in block size; thus they found that for large enough blocks *the block propagation times grew linearly with respect to the block size.*

At the time of their measurement, the average block size was 87KB. This means that back in 2012, it would have taken 5 minutes for 90% of the nodes to receive a full 1MB block — a significant fraction of the block interval.

Since nodes' bandwidth provisioning and the network topology have evolved since 2012, we repeated their measurement recently in 2014 and 2015. Our measurement indicates that the 10%, median, and 90% block propagation times are 0.8 seconds, 8.7 seconds, and 79 seconds respectively. Further, the average block size is now roughly 540KB. Projecting to a 1MB block size, the 90%, median, and 10% block propagation times would be 2.4min, 15.7sec, and 1.5sec respectively.

X% effective throughput. We define the metric "X% effective throughput" as X% effective throughput := (block size)/(X% block propagation delay).

Our measurement study suggests the following X% effective throughputs for the network (with translation to transactions/sec for 250-byte transactions):

X%	**X% effective throughput**	**translated to transactions/sec**	*c.f.* Visa
50%	496 Kbps	248 tx/sec	2000 tx/sec
90%	55 Kbps	26 tx/sec	

We additionally performed experiments to determine the minimum block size for throughput to dominate over latency, similarly mirroring the 2012 study by Decker and Wattenhofer [20]. The results are depicted in Figure 1. That figure shows the overall rate ("network propagation rate") at which blocks propagated to 50%, 75%, and 90% of nodes, capturing the combined effects of latency plus throughput. As block sizes grow from zero, the network propagation rate increases until it levels off at roughly 80KB, suggesting that when the block size is above 80KB, throughput dominates over latency. At this point, the propagation rate for 90% of nodes is about 55 Kbps, which is consistent with the 90% effective throughput observed in the table above for the current overlay network.

As the data are noisy, we do not provide an estimate of the latency. As it is negligible for blocks of size above 80KB, we disregard it elsewhere in the paper.

3.2 Limits of Scalability by Reparametrization

We now explore the potential of reparametrization to scale Bitcoin. If the average block size reaches the X%-effective capacity during a block interval, then (100-X)% of the nodes on the network would be unable even to receive blocks as they arrive, and thus would be effectively disabled.

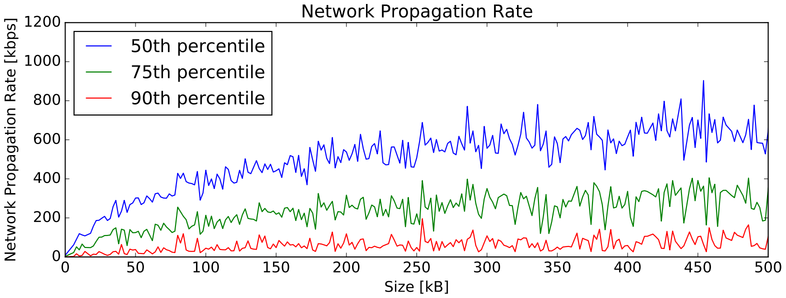

Fig. 1: Network propagation rate (capturing both latency and throughput) vs. block size.

We assume that it is desired to maintain nearly the current degree of decentralization, as measured by the number of properly functioning nodes in the peer-to-peer overlay network. For the purpose of our study, we take 90% to be our target. It is difficult to quantify what each node contributes towards the overall virtue implied by decentralization. Not all nodes are necessarily miners; and while some nodes can be associated with service providers and individual users, there is no absolute measure of the economic significance of each node. Our definition reflects an equal weighting of each node.

We also stress that our results assume the use of Bitcoin's current peer-to-peer overlay network. If the size or properties of the network change, that would affect the $X\%$ effective throughput. We note, however, that Bitcoin's overlay network has remained stable in size from Nov. 2014 to Nov. 2015 at between ∼4500 to ∼6300 full nodes [5]. For brevity, we use the term "current overlay network" to refer to these assumed conditions.

Throughput limit. We observe that the block size and interval must satisfy:

$$\frac{\text{block size}}{\text{X\% effective throughput}} < \text{block interval}.$$

Consequently, for a 10 minutes (or shorter) block interval, the block size should not exceed 4MB for X=90%; and 38MB for X=50%.

Observation 1 (Throughput limit) *Given the current overlay network and today's 10 minute average block interval, the block size should not exceed 4MB. A 4MB block size corresponds to a throughput of at most 27 transactions/sec.*

Latency limit. To improve the system's latency, we can in principle simply reduce the block interval. To do so while retaining high effective throughput, however, would also require a reduction in the block size. Our experiments reflected in Figure 1

Propagating a block of size smaller than 80KB would not make full use of the network's bandwidth, as latency would still be a significant factor in the block's propagation time. To propagate a 80KB block to 90% of the nodes would take roughly 12 seconds, given that the 90% effective throughput of the network today is 55Kbps. Thus the following guideline.

%	**Max. BW** Mbps	**Tx thruput** tx/sec
90%	3.03	**758**
50%	33.03	8.3K
10%	186.10	46K

(a) Lower bound on per-node provisioned bandwidth: measuring 4565 Bitcoin nodes.

Resource	**Max. thruput [1000 tx/sec]**
Tx validation on a modern quad-core processor (2 signatures per Tx)	4
Disk I/O	
Rotational 100MB/sec	200
SSD 300MB/sec	600

(b) Maximum throughput analysis.

Table 2: Per-node resource and bottleneck analysis.

Observation 2 (Latency limit) *Given today's overlay network, to retain at least 90% effective throughput and fully utilize the bandwidth of the network, the block interval should not be significantly smaller than 12s.*

How to interpret/use these numbers. We stress that the above are *conservative bounds* on the extent to which reparametrization alone can scale Bitcoin's peer-to-peer overlay network given its current size and underlying protocols. Other, more difficult-to-measure metrics could also reveal scaling limitations. One example is *fairness.* Our measurement results (see Section 3.1) suggest that in today's Bitcoin overlay network, when nodes are ordered by block propagation time, the top 10% of nodes receive a 1MB block 2.4min earlier than the bottom 10% — meaning that depending on their access to nodes, some miners could obtain a significant and unfair lead over others in solving hash puzzles. Due to complicating factors, e.g., the fact that many miners today do not rely on a single overlay node to obtain transactions (and indeed often rely on a separate, faster mining backbone to propagate blocks), we believe that this figure cannot directly inform reparametrization discussions. It is illustrative of other metrics, however, that may be important but difficult to measure. Consequently, until the Bitcoin system undergoes fundamental protocol changes, gradual or conservative parameter changes may be prudent. Finally, note that our throughput guidelines apply whether parameters are determined by market outcome or enforced by hard-coded limit.

3.3 Bottleneck Analysis

While scaling the blockchain protocol by parameter tuning is possible, we find that best achievable throughput is significantly smaller than the limit posed by the underlying infrastructure.

In Table 2a, we show results from our measurement study where we perform a per-node bandwidth measurement to more than 4000 Bitcoin nodes. Table 2a suggests that individual nodes are provisioned with significantly higher network bandwidth than the overall network throughput attained by Bitcoin today — recall that the 90% effective throughput today is 55Kbps (see Section 3.1). The reason why Bitcoin's network stack cannot reach the per-node link bandwidth is likely due to the combination of several factors. For example, each transaction is transmitted twice, first for gossiping the transaction; and then after a block is mined, the newly mined block will be propagated again including all transactions it contains. Moreover, due to lack of pipelining, propagation over multiple overlay hops introduce delay proportional to the length of the path. Finally, Table 2b shows that the cryptographic overheads associated with transaction verification, and disk I/O are unlikely the bottleneck.

4 Rethinking the Design of a Scalable Blockchain

We discuss techniques that will allow blockchains to scale beyond the parameters of today's Bitcoin. They range from incremental changes atop today's decentralized blockchain to more radical redesigns. The goal of this section is not to propose an end-to-end system, but rather to paint the design space, suggest promising approaches, and pose open challenges to the community.

We organize our discussion around a decomposition of the Bitcoin system into a set of abstraction layers that we call *planes*. Ordered in a hierarchy of dependency from bottom to top, the five planes we consider are the Network, Consensus, Storage, View, and Side Planes.

In our exposition here, the *ledger* is the full history of the system, the complete output of the *Consensus Plane*, as we define it below. A more precise definition is possible that also specifies a particular confirmation policy, as ledger contents may be subject to change, as in decentralized cryptocurrencies. For simplicity, we do not model this feature of decentralized cryptocurrencies and instead treat writes to the ledger as confirmed.

4.1 Network Plane

The function of the Network Plane is to propagate transaction messages. It supports a broadcast abstraction in which a transaction message from any player is transmitted to all (full) nodes in the Bitcoin network. The Network Plane in Bitcoin is not a pure broadcast medium, however. Nodes only propagate messages that represent valid transactions and thus the abstraction only accepts valid transactions as inputs.

Our measurements have shown that Bitcoin's network protocol and implementation do not fully utilize underlying network bandwidth, making Bitcoin's Network Plane the bottleneck in transaction processing. A natural direction to improve scaling in Bitcoin is thus to improve the design of its Network Plane.

Two inefficiencies in Bitcoin's Network Plane stand out. First, to avoid denial-of-service by propagation of invalid transactions, a node must fully receive and validate a transaction before further propagating it. (To be deemed valid, a transaction must ingest and produce legitimate transaction outputs and not conflict with previous transactions.) This local validation of transactions contributes significantly to the overall propagation time. Second, Bitcoin's network-layer protocol first propagates all transactions, and then propagates a block (containing previously propagated transactions) again when it is mined. This effectively requires each transaction to be transmitted twice.

There have been several proposals to improve Bitcoin's network-layer protocol. One possibility, to avoid transferring each transaction twice, is to rely on a set reconciliation protocol in which nodes only fetch transactions that they do not possess in a newly mined block [6, 31, 38, 47]. Another option, in use by miners today, is to use a dedicated, centralized, high-speed relay network for inter-miner communication [19].

A different direction is to improve the network layer's function as a broadcast channel. The Network Plane could be designed as a robust P2P overlay topology, with strong connectivity between honest nodes and a low diameter. Such overlay topologies are usually expander graphs, which have known low-latency broadcast protocols [28, 32]. To limit influence of adversarial nodes, the overlay could randomize the location of all peers (outside of their control). Several previous distributed systems have adopted this approach [14, 43]. To further limit denial-of-service, nodes could rate-limit transmissions from their peers. Designing such an overlay which maintains a strong connectivity between honest nodes (in the presence of byzantine adversaries) is well-known for static networks [22, 35], but for highly dynamic networks is an active area of research [30].

A longstanding issue involving the Network Plane is the incentivization of the participants. Researchers have noted that Network Plane lacks an incentive structure for the dissemination of transactions, and have proposed a modified fee splitting structure to provide robust incentives. Many other aspects of the network protocol rely on voluntary participation and require ad hoc defenses to stem flooding and denial of service attacks.

4.2 Consensus Plane

The function of the Consensus Plane is to designate a globally accepted set of transactions for processing, as well as a total or partial order on these transactions. As a general abstraction, this plane ingests messages from the Network Plane and outputs transactions for insertion into the system ledger. In Bitcoin, the Consensus Plane is the functionality that mines blocks and reaches consensus on their integration into the blockchain.

Improving proof-of-work protocols. Bitcoin's blockchain protocol introduces a tradeoff among consensus speed, bandwidth, and security. By improving the former two, one introduces an increased number of forks, leading to a loss of the mining power that secures the system and to reduced fairness [25]. Many cryptocurrencies (e.g. [9]) favor consensus speed over security, employing a standard Bitcoin blockchain with a high block-generation frequency.

This three-way tradeoff, however, is not inherent in decentralized cryptocurrencies. The GHOST protocol [45] of Sompolinsky et al. as well as Lewenberg et al. [36] demonstrate that fairness and mining power utilization can be improved by changing the chain selection rule, in particular, by being inclusive to forks outside the main chain as well. In more recent work, Bitcoin-NG [25] demonstrates that the inherent tradeoffs in Bitcoin can be eliminated with an alternative blockchain protocol, offering a consensus delay and bandwidth limited only by the Network Plane.

Proof of stake. Various proposals (e.g. [11, 33]) use *proof of stake* to achieve consensus, eliminating the computational expense of proofs of work. In proof of stake, principals gain the right to create blocks by depositing funds they own. These techniques, however, lack formal guarantees of system convergence [18].

Consortium consensus. Decentralization carries a performance cost. A trust model with stronger assumptions than those in Bitcoin can support a more efficient consensus protocol, achieving better latency and throughput with less computation, bandwidth, and storage. Specifically, using a standard Byzantine Fault Tolerant (BFT) replication protocol with a small number of pre-designated trusted entities removes many of the scaling obstacles in Bitcoin.

Settings involving BFT protocols executed by small sets of trusted entities have received little treatment in the academic literature, but are of considerable interest in practice, and mainstream financial institutions are actively exploring their use [44]. They are sometimes referred to as "consortium blockchains."

Consortium blockchains are worth investigation both as an alternative to decentralized cryptocurrencies and to characterize the performance cost that decentralized blockchains incur by distributing trust. In Appendix A, we present performance figures and microbenchmarking results on experiments with a popular BFT protocol (PBFT) across a range of different system parametrizations and with nodes dispersed across eight geographies worldwide. Our results illustrate the attractiveness of BFT as a basis for the consensus layer in a cryptocurrency (given acceptance of its strong trust assumptions). Even with dozens of nodes, PBFT greatly outperforms Bitcoin in both transaction latency and throughput. For example, 64 nodes processing batches of 8192 transactions can achieve a throughput of **4.5K tx/sec**., average transaction latency of **1.79 sec**., and an estimated resource cost per transaction of just $\mathbf{\$3.95 \times 10^{-7}}$.[2] Scaling to hundreds of nodes, however, would greatly degrade the performance of the system. As we now explain, a promising approach to scaling and an open research direction is how to *shard* a BFT protocol.

Sharding. One possible technique for improving the scalability of the Consensus Plane is to shard it, that is, split up the task of consensus among concurrently operating sets of nodes, with the aim of improving throughput and reducing per-node processing and storage requirements. Sharding is commonly employed in distributed databases, such as Dynamo, MongoDB, MySQL, and BigTable, although performance typically does not grow linearly with shard count. This is due to the need to reach consensus among the shards when operations span multiple shards. One possibility, explored in a non-Byzantine environment in past work [24, 29, 51], is to use a separate consensus protocol, such as Paxos, to achieve agreement among the shards. Such schemes, however, can incur substantial overhead when cross-shard coordination is required in a Byzantine setting, so sharding protocols for blockchains are an open area of research.

[2] Transactions in our experiments are 190 bytes long, all that is needed for a basic money transfer; given the roughly 500 byte average size of Bitcoin transactions, the system would achieve 1.7k tx/sec.

Delegation of trust and a hierarchy of sidechains. Another technique for scaling is to create a hierarchy of lower-tier "consensus instances," commonly referred to as "sidechains." Sidechains can potentially have a lower degree of decentralization than the top-level blockchain. Sidechains may also run non-proof-of-work consensus protocols, such as BFT. One sidechain structure, proposed by Back et al. [15], permits transactions to move funds among independent chains.

The introduction of sidechains raises three technical challenges. First, the sidechains must be secured independently of the main blockchain. Merged mining techniques [15] allow separate chains to share their mining power, but require miner coordination. Without such coordination, the maintenance of sidechains dilutes the mining power in the system, rendering the individual chains vulnerable. Second, if sidechains are widely adopted, the chances that a given source of funds and a desired destination are on the same sidechain are small, requiring inter-chain transactions. Inter-chain transactions will have to go through the main blockchain, possibly requiring two separate transactions, and therefore may place more of a burden on the main blockchain and thus have an adverse impact on scalability. Finally, transactions involving more than one decentralized chain may incur high latency. Decentralized blockchains require the accumulation of a number of blocks to ensure that a transaction will remain in the blockchain with high probability. Transactions among chains will require a sequence of such block accumulations, one per chain.

4.3 Storage Plane

The Storage Plane functions as a global memory that stores and provides availability for authenticated data produced by the Consensus Plane. It may be regarded as an abstraction with two interfaces: (1) It ingests and processes memory-modification instructions—`write` and (potentially) `delete` operations—from the Consensus Plane and (2) It services `read` requests from any entity in the system. The storage plane contains the ledger of the system but may also contain other state produced by consensus, such as smart contract state or "views" supported by the View Plane.

There are several ways to implement the Storage Plane in a cryptocurrency. In Bitcoin, the Storage Plane may be regarded as storing the Bitcoin ledger. The Bitcoin reference implementation today by default stores the entire ledger; as a result, the system stores many duplicates of the entire ledger. The Storage Plane in Bitcoin accepts only `writes` that append data, namely newly mined blocks, and does not support `delete` operations. The only generally supported `read` operation for the Bitcoin Storage Plane downloads the contents of the entire ledger, a process that requires four days (see Section 2). (Given the current height of the blockchain, a downloaded ledger may be authenticated by reference to the genesis block.) Thus, Bitcoin's Storage Plane has notable inefficiencies.

Other implementations of and interfaces for the Storage Plane are possible. The community has proposed interesting ideas that can essentially shard the storage of a UTXO data structure [37] (see below). It is not clear how these ideas would generalize to other forms of state that might go into the Storage Plane, e.g., the state associated with smart contracts. How to shard a general-purpose Storage Plane such that not all consensus nodes have to store it in its entirety and such that its contents can be authenticated during `read` operations is an open research challenge. Distributed Hash Tables (DHT) are a possible start, coupled with suitable data authentication techniques.

4.4 View Plane

For Bitcoin miners, it is unnecessary to operate on the full ledger that stores the entire transaction history. Thus miners and nodes in Bitcoin locally compute and operate on a view of the ledger called the *unspent transaction outputs* (UTXO) set, which in effect specifies the current balance of all entities in the system. Similarly, in Ethereum smart contracts can define state that resides in the ledger. Parties to a smart contract may wish to access and authenticate this state without reading other parts of the ledger. For this reason, a key performance requirement in cryptocurrencies (both decentralized and probably centralized) is support for *views*.[3]

A view is a data structure derived from the full ledger whose state is obtained by applying all transactions. For performance reasons, a view may be stored in the Storage Plane and distributed in an authenticated fashion — Bitcoin did not implement this optimization, and therefore new miners would now need four days to reconstruct the UTXO set (which can be considered as a view) from the beginning of time. In general, the view can be an arbitrary function of the full ledger, not necessarily the UTXO set. As a piece of data in the Storage Plane, a view must be determined either implicitly or explicitly by the Consensus Plane and must be authenticable by any entity executing a Read operation against it. There are a number of options for implementing a view, including the following.

Views via replication. Bitcoin [39], Ethereum [49], and other popular decentralized cryptocurrencies require all consensus nodes to verify all transactions (and/or smart contracts), and based on the result of the computation, update their respective views, e.g., UTXO sets, locally. In this case, the view is an implicit output of the Consensus Plane and may be regarded as residing in the Storage output: provided that it is correctly computed, it represents the computation that an honest set of consensus nodes would produce, and of course it has high availability.

Outsourcing views via cryptography. It is possible to outsource the computation of a view to a third-party service provider. This provider may release a cryptographic digest (e.g., Merkle-tree root) of this view along with a proof of its correctness. By relying on verifiable computation techniques such as succinct non-interactive arguments of knowledge (SNARKs) [17, 40], the provider can produce a proof of correctness for the digest, supporting authentication of the view. The view may then be inserted into the Storage Plane. If availability is not essential, the view can alternatively be stored within some other part of the system, e.g., by the provider itself, rather than in the Storage Plane.

[3] We adapt the term "view" from its meaning in database theory, where it refers to the result set of a stored query.

One advantage of this approach is that consensus nodes now need not store the entire ledger. They can instead operate over suitably chosen views. A key question that must be answered, however, is whether cryptographic techniques such as SNARKs are practically viable. In the Appendix, we present experimental results showing that the amortized cost of employing SNARKs can be as low as **$0.0154** per transaction for computing a simple view that essentially stores all users' balances.

4.5 Side Plane

Much as sidechains allow off-the-main-chain consensus, we can consider off-chain functionalities. Off-chain transactions have been demonstrated in payment networks [12,21,41], in which payments are routed along paths of pre-established "collateral" channels. Each such channel represents a quantity of bitcoin reserves set aside, such that parties can repeatedly adjust their relative stake by exchanging out-of-band messages until the channel is finalized (and the reserves paid out). While payment networks have been heralded as a solution to Bitcoin's inherent limitations, much of their operation, and the guarantees they can offer, rely critically on the nature of the links formed between parties. Even when payment networks use the same underlying transaction format as Bitcoin, as do the Lightning Network [41] and full duplex channels [21], they essentially form a separate Network Plane as well as an independent, peer-to-peer Consensus Plane, backed by Bitcoin. As a result, their capacity, ability to find routes, achieved throughput, latency, and privacy guarantees depend fundamentally on emergent properties of the payment network graph, such as the value capacity of peer-to-peer channels, the discoverability of routes, the online status of nodes involved, and so on. Further, payment channels may embody a similar tradeoff between performance and centralization in the payment network; a centralized hub-and-spoke topology that simplifies routing embodies inherent problems with centralization, such as loss of privacy. The design of protocols for efficient, scalable, privacy-preserving payment networks is an ongoing area of research: it is far from a given that they can outperform Bitcoin's Network and Consensus layers overall.

5 Conclusion

This paper explored the challenges in scaling Bitcoin and blockchains in general. Supported by measurement studies, we showed that reparametrization of the block size and interval in Bitcoin is only a first step toward substantial throughput and latency improvements while retaining significant system decentralization. More aggressive scaling will in the longer term require fundamental protocol redesign. Through a structured presentation of the design landscape for blockchain protocols, we illustrated the variety of potentially successful approaches to such scaling, categorized a range of recently proposed and new ideas, and framed a number of important open technical challenges for the community.

Acknowledgements

This work is supported in part by NSF grants CNS-1314857, CNS-1453634, CNS-1518765, CNS-1514261, CNS-1518899, a Packard Fellowship, a Sloan Fellowship, two Google Faculty Research Awards, and a VMWare Research Award.

References

1. https://en.bitcoin.it/wiki/Scalability.
2. https://en.bitcoin.it/wiki/Mining_hardware_comparison.
3. https://blockchain.info/charts/hash-rate.
4. https://blockchain.info/charts/n-transactions-per-block.
5. https://bitnodes.21.co/dashboard/?days=365.
6. https://gist.github.com/gavinandresen/e20c3b5a1d4b97f79ac2.
7. Amazon ec2 pricing. http://aws.amazon.com/ec2/pricing/. accessed 2015-10-30.
8. Antminer s5+ hardware. https://bitmaintech.com/productDetail.htm?pid=00020150814075326555O4JMKzsM067B. accessed 2015-10-30.
9. Litecoin, open source P2P digital currency. https://litecoin.org.
10. How a Visa transaction works. http://web.archive.org/web/20160121231718/http://apps.usa.visa.com/merchants/become-a-merchant/how-a-visa-transaction-works.jsp, 2015.
11. NXT.org, Decentralized Financial Ecosystem. http://nxt.org/, 2015.
12. abhi shelat and R. Pass. Micropayments for peer-to-peer currencies. In *CCS*, 2015.
13. G. Andresen. Increase maximum block size (bip 101). https://github.com/bitcoin/bips/blob/master/bip-0101.mediawiki, retrieved October 2015.
14. B. Awerbuch and C. Scheideler. Towards a scalable and robust DHT. In *SPAA*, 2006.
15. A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. Enabling blockchain innovations with pegged sidechains. https://www.blockstream.com/sidechains.pdf, retrieved Nov 26, 2015.
16. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*. 2014.
17. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *Security*, 2014.
18. I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. https://eprint.iacr.org/2014/452/.
19. M. Corallo. High-speed Bitcoin relay network. https://github.com/TheBlueMatt/RelayNode, December 2015.
20. C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P*, pages 1–10. IEEE, 2013.
21. C. Decker and R. Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18. Springer, 2015.
22. S. Dolev and N. Tzachar. Spanders: Distributed spanning expanders. *Sci. Comput. Program.*
23. P. W. Eric Lombrozo, Johnson Lau. Segregated witness (consensus layer). https://github.com/CodeShark/bips/blob/segwit/bip-codeshark-jl2012-segwit.mediawiki.
24. R. Escriva, B. Wong, and E. G. Sirer. Warp: Lightweight Multi-Key Transactions for Key-Value Stores. http://arxiv.org/abs/1509.07815.

25. I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. Technical report, CoRR, 2015.
26. J. Garzik. Block size increase to 2mb (bip 102). `https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki`, retrieved October 2015.
27. J. Garzik. Making decentralized economic policy. `http://gtf.org/garzik/bitcoin/BIP100-blocksizechangeproposal.pdf`, retrieved October 2015.
28. C. Georgiou, S. Gilbert, R. Guerraoui, and D. R. Kowalski. Asynchronous gossip. *J. ACM*, 60(2), May 2013.
29. L. Glendenning, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Scalable Consistency in Scatter. In *SOSP*, 2011.
30. R. Guerraoui, F. Huc, and A.-M. Kermarrec. Highly dynamic distributed computing with byzantine failures. In *PODC*, 2013.
31. H. D. Johansen, R. V. Renesse, Y. Vigfusson, and D. Johansen. Fireflies: A secure and scalable membership and gossip service. *ACM Trans. Comput. Syst.*, 2015.
32. R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *FOCS*, 2000.
33. S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, August 2012.
34. A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, abhi shelat, and E. Shi. How to use snarks in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. `http://eprint.iacr.org/`.
35. C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *In IEEE Infocom*, pages 2133–2143, 2003.
36. Y. Lewenberg, Y. Sompolinsky, and A. Zohar. Inclusive block chain protocols. In *FC*, 2015.
37. G. Maxwell. `https://bitcointalk.org/index.php?topic=314467#msg3371194`.
38. Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. In *IEEE Trans. on Information Theory*, 2003.
39. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. `http://bitcoin.org/bitcoin.pdf`, 2009.
40. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *S & P*, 2013.
41. J. Poon and T. Dryja. The bitcoin lightning network. `https://lightning.network/lightning-network-paper.pdf`, retrieved Nov 26, 2015.
42. P. Rizun. A transaction fee market exists without a block size limit. 2015.
43. S. Sen and M. J. Freedman. Commensal cuckoo: Secure group partitioning for large-scale services. *SIGOPS Oper. Syst. Rev.*, 2012.
44. L. Shin. Bitcoin blockchain technology in financial services: How the disruption will play out. *Forbes*, 14 September 2015.
45. Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In *FC*, 2015.
46. TradeBlock. Bitcoin network capacity analysis. `https://tradeblock.com/blog/bitcoin-network-capacity-analysis-part-1-macro-block-trends`.
47. R. van Renesse, D. Dumitriu, V. Gough, and C. Thomas. Efficient reconciliation and flow control for anti-entropy protocols. In *LADIS*, 2008.
48. L. Wilson. Average electricity prices around the world. `http://shrinkthatfootprint.com/average-electricity-prices-kwh`.
49. G. Wood. Ethereum: A secure decentralized transaction ledger. `http://gavwood.com/paper.pdf`, 2014.
50. P. Wuille. Block size following technological growth (bip 103). `https://github.com/bitcoin/bips/blob/master/bip-0103.mediawiki`, retrieved November 2015.

51. C. Xie, C. Su, C. Littley, L. Alvisi, M. Kapritsos, and Y. Wang. High-Performance ACID via Modular Concurrency Control. In *SOSP*, 2015.

Table 3: **Consortium blockchain scalability.** Results of running PBFT over geographically distributed EC2 nodes for some representative (n, k) parameterizations. As for any BFT protocol, the throughput drops as the number n of nodes increases. Conversely, in this small experiment, enlarging the batch size k may be seen to increase throughput at the cost of a small increase in latency.

# of Nodes (n)	Batch size (k)	Latency	Throughput	Price per tx	500 byte tx
4 nodes (1 region)	32768	288 ms	113K tx/sec	\$9.83 $\times 10^{-10}$	42.9K tx/sec
8 nodes	8192	0.58 sec	14.0K tx/sec	\$1.59 $\times 10^{-8}$	5.3K tx/sec
8 nodes	32768	1.48 sec	22.2K tx/sec	\$1.00 $\times 10^{-8}$	8.4K tx/sec
16 nodes	8192	0.69 sec	11.9K tx/sec	\$3.73 $\times 10^{-8}$	4.5K tx/sec
16 nodes	16384	1.04 sec	15.8K tx/sec	\$2.81 $\times 10^{-8}$	6.0K tx/sec
32 nodes	2048	0.48 sec	4.3K tx/sec	\$2.07 $\times 10^{-7}$	1.6K tx/sec
32 nodes	8192	0.925 sec	8.8K tx/sec	\$1.01 $\times 10^{-7}$	3.3K tx/sec
64 nodes	2048	0.824 sec	2.4K tx/sec	\$7.40 $\times 10^{-7}$	0.9K tx/sec
64 nodes	8192	1.79 sec	4.5K tx/sec	\$3.95 $\times 10^{-7}$	1.7K tx/sec

A BFT Experiments (Consortium Consensus)

Here we report on our experiments on BFT performance. Table 3 illustrates the attractiveness of BFT as a basis for the consensus layer in a cryptocurrency (given acceptance of its strong trust assumptions). Even with dozens of nodes, PBFT greatly outperforms Bitcoin in both transaction latency and throughput. Scaling to hundreds of nodes, however, would greatly degrade the performance of the system.

Experiments were conducted using t2.medium Amazon EC2 instances. The results are displayed in Table 3. The 4 node experiment represents a best-case setting; all nodes were located in the US East N. Virginia region. In all other experiments the nodes, plus a client furnishing transaction inputs but not participating in the consensus protocol, were evenly distributed across 8 geographical regions: Northern Virginia, Oregon, Northern California, Ireland, Frankfurt, Tokyo, Sydney, Sao Paulo. As instance costs vary by region, we conservatively assume each node incurs the highest observed fee of \$0.10 per hour. We experimented with several batch sizes in order to locate the point at which each configuration becomes bandwidth bound. Here, a transaction message is 190 bytes in length and is modeled after a Bitcoin transaction, including a pay-to-public-key-hash scheme and a digital signature.

We observed that configurations with a larger number of nodes became bandwidth bound at smaller batch sizes. This is due to the primary node broadcasting each batch to every other node participating in the protocol. Increasing the batch size beyond this bound for a given configuration increases latency while leaving throughput relatively unchanged. For smaller batch sizes the bottleneck was local processing, primarily signature verification for the batch as a whole.

Table 4: Verifiable outsourcing of ledger storage and maintenance using SNARKs

Rate tx/block	Total Proof time (prover)	Verification time (verifiers)	Proof Size	# of EC2s	Total Cost (prover)	Cost per tx (prover)
25	496 sec	0.01 sec	288 bytes	1	$0.385	$0.0154
4200 (7tx/sec)	588 sec	1.68 sec	48 kbytes	168	$64.68	
6000 (10tx/sec)	628 sec	2.4 sec	68 kbytes	240	$92.40	

Once an ordering of batches has been agreed upon and committed, individual transactions may be processed and verified independently from the consensus protocol. However, we expect to see a much larger slowdown when scaling to hundreds of nodes, as the number of messages in standard BFT protocols grows quadratically in the number of nodes.

B Use of SNARKs for Outsourcing View Computation

We now explain how SNARKs may be used to support computation of views by a service provider, rather than individually by every node in the network (as happens today). In our experiment, we assume that a simple view is adopted containing all users' account balances.

To support the secure outsourcing of view derivation, the provider (or prover, as we might call it) must store the balance for each Bitcoin address used in the transactions so far. As in Bitcoin, we assume 2^{160} possible public addresses. The prover will maintain a Merkle tree of height 160, where each leaf stores the balance of the public address identified by the path to the leaf. (Unused zero-balance leaves do not have to be stored explicitly.) Computing the initial digest of the tree when all balances are zero can be done by utilizing the similarity across levels. Later, when a transaction is received, the system checks the balance of the sender address, and if it is sufficient, transfers the desired amount to the receiver's balance. (In case of a mining reward, the initial check is not necessary.) The prover then updates the ledger digests accordingly.

Using SNARKs, the service provider will be able to prove the correct application of a set of transactions, and the correct update of digests. We evaluated a SNARK circuit that checks and applies 25 1-sender 1-receiver transactions to the ledger. Each transaction in that case specifies the desired amount of money to be transferred. This is equivalent to 1-input 2-output transactions in Bitcoin, where one of the outputs is a remainder going back to the sender's address, but the remainder does not have to be specified explicitly in our setting. The prover then outputs the modified digest with a vector representing which transactions were valid, and possibly the modified balances. To make the circuit efficient, we used a SNARK-friendly collision resistant function based on subset sum [16], at 80-bit of security as in [34]. To experiment at higher transaction rate, multiple circuits can be run in a nearly-parallel mode, by computing and feeding the next digest quickly from one circuit to the next one without waiting for the proof to complete.

Table 4 shows the estimated cost incurred by the prover (provider) and the verifiers (relying miners or consensus nodes) to produce the proof at multiple settings. We ran the experiment for the first case using an Amazon EC2 r3.8xlarge instance [7] (using 32-cores for the proof computation, and single core for verification) and using libsnark [17] as a backend, and estimated the overall performance and cost accordingly assuming throughputs of 7 tx/sec, and 10 tx/sec. The table shows that the computation cost for applying one transaction to the ledger is about $0.0154, but at high throughput, the waiting time for proof computation is about or more than 10 minutes (implying a higher delay if higher security level is used).

By adopting SNARKs, miners no longer need to store the full ledger nor the necessary views to validate transactions. In this way, decentralized storage and replication of the ledger and views can be decomposed from the consensus protocol and the view computation. Additionally, in our current experiments, we assume that the consensus nodes are validating the signatures on the transactions, and this signature validation is not performed within SNARKs (otherwise the cost of SNARKs would be more expensive).