Experimental Comparison of Blockchain Scaling Protocols Using Virtual Machines

Michael Lin

Introduction

The Internet is an instrumental part of modern communication and commerce, with approximately 4 billion users connected concurrently in the entire world, however, connections suffer from unreliability, faults, and attacks (Kemp, 2018). Tools such as cryptography (the algorithmic control of access to information) and redundancy in the design of computer systems serving clients were implemented gradually over time (Rescorla, 2018). The goal of these technologies was to allow users, both clients and servers alike, to agree that they share the same information and that nothing was altered before, during or after transport. These technologies are still evolving, meaning that data agreement may occasionally fail. This is well exemplified by situations involving disputed transactions between financial institutions such as banks, and the clients they service (customers and merchants). First introduced by the currency protocol Bitcoin, the "blockchain" is a type of data structure that can only be appended to, and not erasable. It finds many uses, most commonly to transact currency in a provably fair method that involves no intermediate bank or payment network. Arising from its simplistic nature of appending collections of transactions to its end periodically as "blocks", blockchains operate by code only and are decentralized, meaning they are run and maintained by user's computers rather than that of an institution. Currently, blockchains designed for public use cannot scale to serve as many users compared to a major credit card/payment network without sacrificing basic security and suffering attacks by malicious users.

Review of Literature

Digital Authentication

The concept of digital signatures was first introduced when the RSA cryptosystem was published (Rivest *et al.*, 1978). RSA was an asymmetric algorithm that accomplished two goals:


"asymmetric" data privacy and verification. RSA allowed users to encrypt and send data that only the recipient can decrypt. On top of this, users can verify data that only the issuer of a message can sign (a signature is generated). Two mathematically related numbers, called public and private keys, can be used for encryption/verification and decryption/signing, respectively. RSA, though still in use today, was an inefficient algorithm. RSA's security is derived from the computational hardness of factoring a large composite number. This is referred to as the integer factorization problem. Due to its reliance on very large semiprime numbers (two large prime numbers multiplied together) for its security, key pair sizes had to grow in large increments to account for increasingly powerful computers to stay secure. In turn, ciphertext (encrypted data) and signature size had to grow along with the increasing key size due to the nature of RSA. The concept of using elliptic curves, a type of mathematical curve that also has a computationally hard problem called the elliptic curve discrete logarithm problem (ECDLP), to replace RSA and similarly designed schemes gained popularity in recent years. Elliptic curve schemes could do everything that RSA could do with smaller keys (Koblitz, 2001). Simply put, by finding the tangent of a point on an elliptic curve and reflecting across the *x*-axis to another point on the curve and repeating, it is hard to find the number of times this operation was performed given only the starting and ending points (see Figure 1). Elliptic curves allowed the same operations of RSA but with smaller keys, signatures, and ciphertexts. For example, an

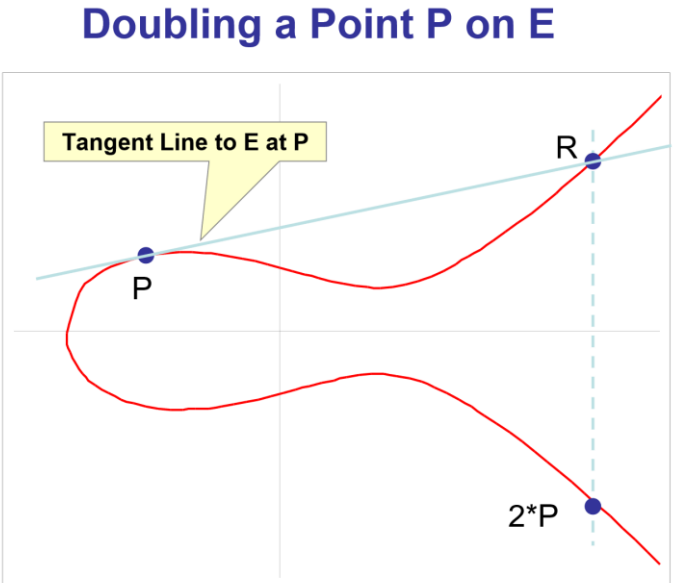

Figure 1. Illustration of a single point multiplication of point P. In practice, many multiplications are used, and the number of multiplications is the private key K, and the public key is the end point. The originating point is referred to as the generator G. (Srinivasan, 2011)

RSA-3072 key, which is a secure key size at the time of writing, is equivalent to a 256-bit public key. Replacing RSA with elliptic curves meant less space consumption in large databases. While the difference may seem small, signature data can be reduced by a factor of ~10 times on large datasets. Despite their differences, both algorithms employ the use of a cryptographic hash function, or simply "hash functions". Hash functions take any input data and return a fixed length, seemingly random number. Given this number, it is hard to find the original data or other data with the same hash, however, hashing itself is trivial. Modern security protocols sign hashes rather than raw data. They can be abstractly thought of as a "unique labeling program", no two distinct messages can be found with the same label and knowing the label of one message tells nothing about the label of a very similar message. It is, therefore, safe to sign a "label" of a message rather than the message itself.

As convenient as elliptic curves were, generating digital signatures and decrypting data is inherently slower on elliptic curve algorithms compared to RSA of equivalent security. If a user wanted to check the authenticity of some message in a set (say, a list of transactions between bank accounts), the flawed and inefficient approach of the set provider would be to sign each element of the set, which takes a long time to sign (especially for larger datasets), as well as more storage space. However, by using a cascade of hashes in a "tree fashion" (see *Figure 2*, below "Block 11"), where hashes are themselves hashed together to form a single hash at the top of the tree, any piece of data could be verified by providing only a few hashes and a single signature. This is very efficient due to hashes being much smaller than signatures, with an added benefit that messages considered expired or no longer useful, e.g. very old transactions from frozen bank accounts, can be deleted so long as the required hashes are kept (Merkle, 1979).

Blockchains

Using ECDSA, an elliptic curve signature algorithm, as well as the tree construction, called "Merkle trees", blockchains were first used to store transactions between accounts holding currency in a protocol known as Bitcoin, created as a transparent and peer-to-peer alternative to traditional banking (Nakamoto, 2008).

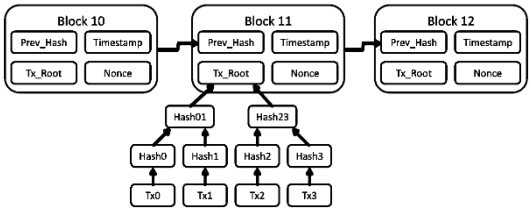

Figure 2. Abstract flow chart showing transactions, a Merkle tree, and blocks, and how they are hashed together
(Blockgeeks, 2017)

Besides simple financial transactions, blockchains could perform more complex operations. A blockchain protocol called Ethereum was created for that purpose (Wood, 2018). This blockchain relies on the principles of "smart contracts", decentralized pieces of code (Szabo, 1997). These contracts are Turing-complete, meaning they can perform any operation a regular computer can do, such as holding currency for organizations, creating securities, executing escrows between accounts, etc., all without trusting a central institution to secure code execution. Unlike legal contracts which are enforced by courts, smart contracts are self-enforcing; it is impossible for any "terms" to be breached because all computation is subject to strictly codified rules and all computers running the smart contract will reach the same result, preventing any possible disputes.

The Scalability Trilemma

The Scalability Trilemma, a set of three desirable properties: security, scalability, and decentralization, is a common problem for all public distributed systems, especially blockchains. Security defines the difficulty of an attacker to gain control of a blockchain in proportion to the number of computers/nodes running it (the more computers, the harder it is to attack), scalability defines the number of transactions or operations that a blockchain can handle before becoming unacceptably congested (typically measured in transactions per second, or TPS), decentralization

defines how much of the network is controlled by actual users as opposed to centralized nodes (such as light client service providers, which provide transaction data for "light clients" that cannot download entire blockchains, such as mobile phones). Bitcoin is inherently decentralized due to its "gossip protocol" (the spreading of blocks and transactions in a web-like fashion across a network) from the beginning, but its price and user base grew since it first started. To generate secure blocks that are hard to reverse, a hash-based "Proof-of-work" (PoW) must be calculated for every additional block, which allows transactions to be added to the blockchain. This is effective at securing the blockchain as it prevents disputes as to which blocks are canonical. Blocks with valid PoW require a provable amount of electricity in order to generate, which itself costs money, indicating financial incentive in providing a valid block. Any change in a transaction invalidates PoW, no matter how little, which means that only correct or "canonical" transaction history will be accepted by the network. Each PoW needs a predictable amount of time to calculate. Users that engage in calculating PoW are known colloquially as miners. Since mining costs electricity, miners are rewarded with new bitcoins for generating blocks that further the history of the blockchain. Miners began to design efficient chips to mine faster and consume less electricity. Mining has become more and more difficult (an inbuilt feature of Bitcoin to keep up with progressing technology), leading to network centralization, where only large miners had any realistic chance of profiting. Besides the centralization, users also dealt with network congestion due to the growing user base. Blocks can only be up to 1 megabyte in size, and transactions count towards this quota. Transactions would take very long times to be confirmed by the network, and transaction fees also grew (see *Figure 3*).

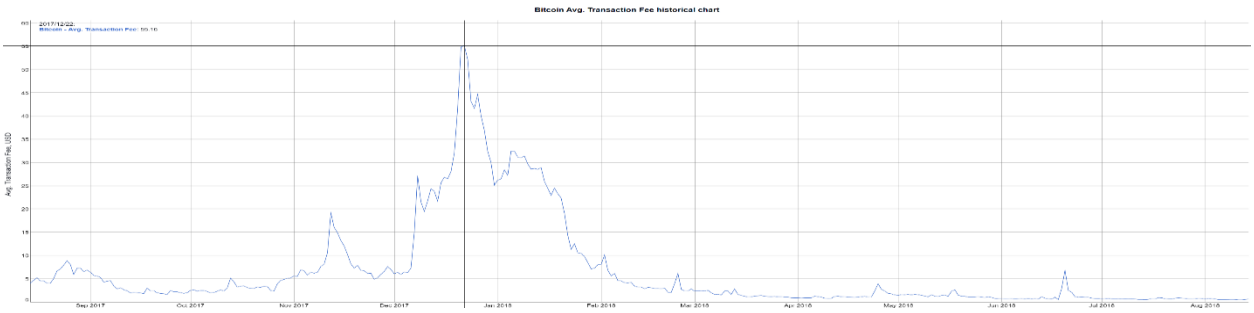

Figure 3. Historically high fees on Bitcoin, peaking at $55.16 USD per transaction. Prices according to exchanges. (BitInfoCharts, 2018)

An early idea to fix the problem was a change in protocol to allow for larger blocks. One hard fork (variant) of Bitcoin, called Bitcoin Cash, implements this idea by increasing block sizes to 8 MB. Theoretically, this would increase the scalability, but it would also increase centralization of blockchains, as larger blocks mean fewer users willing to give up storage space. Another scheme, called "merged mining", allows PoW for Bitcoin to be used on other chains. In reality, this is actually a stealthy method of increasing the block size, as miners have to handle both chains. When blockchains become centralized, they are easier to attack due to fewer nodes on the network, and therefore less for the attacker to need to take over. Attempting to reduce the block time too much also introduced issues in consensus due to the inherent latency of computers on the Internet (Buterin, 2015). Running multiple blockchains to increase scalability without increasing block size was another common practice as well (spawning the colloquial term "altcoin", or alternative cryptocurrency), the more blockchains, however, the less security for each chain. For every change, a tradeoff would always occur.

Problem Statement

- There is little agreement on which protocols if any proposed so far, are most effective at achieving all three properties of the Trilemma. The reason for this is lack of formal experimentation data, rather than lack of theoretical algorithm analysis.

Objective

- Create and demonstrate an experimental model for comparing public blockchains with built-in cryptocurrencies on vanilla (unmodified) Bitcoin as a control for future experimentation.

Methodology

This experiment was scripted and carried out by me, under the guidance of my mentor to troubleshoot technical issues and design problems, as well as providing advice for constants and experiment design. All node images, command OS images, and commanding code was written by me under the GNU Public License 3.0 in Python 3.7.1 syntax, with the use of the Bitcoin Daemon (`bitcoind`) and the library `python-bitcoinrpc` to make JSON remote procedure calls (JSON-RPC), all sourced under the GPL 2.1 (and compatible versions). The total runtime of the virtual machines was 5 days, carried out during Q4 2018, with all small-scale deployment testing with 3 nodes done Q3 2018. All scripts were mostly written in the first half of 2018 and initially checked using preconfigured Docker containers to run RPC against. *Table 1* denotes symbols representing parts of the model. See the master code repository[1] for more details.

[1] GitHub:freerainboxbox/bitcoin-scale-test, master branch

Table 1. Elementary Symbols and Functions

CT_s	Median block time in seconds
CT_b	Minimum number of block confirmations before safety in a selfish mining attack
SafeTime	Minimum time before safety, in seconds — $SafeTime \equiv CT_s \times CT_b$
Tx	A single transaction (determined by context)
Blk	A single block (determined by context)
Blk_α	Genesis block of the applicable blockchain. If initial reward generation (also called a "premine") is required, Blk_α is the block the experiment started.
B	The blockchain, a vector of blocks, which are vectors of transactions, all sorted by time
r	The current instantaneous count of transactions per second
B_r	The blockchain segment of TPS *r*, such that $B_r \in B$
Blk_r	The first block of the TPS segment B_r.
R()	TPS interval of *Blk* or *Tx*.
min()	Returns smallest valued argument of input arguments
max()	Returns largest valued argument of input arguments
mod(*x*,*y*)	Calculate remainder of the Euclidean division of *x* by *y*
Ω()	Best-case scenario (asymptotic)
O()	Worst-case scenario (asymptotic)
BT()	Number of blocks before inclusion of *Tx*, returns a signed integer (int)
BH()	Height of input *Tx* or *Blk*, returns an unsigned integer (uint) or float.
Fee()	Returns the fee of *Tx* in atomic cryptocurrency units, such as satoshis or wei.
T()	Returns the timestamp of *Tx* or *Blk* as seconds past January 1st, 1970, 00:00:00 UTC as a uint.
|*x*|	Returns the cardinality of set/tuple/vector *x*. Not to be confused with absolute value.
⌈*x*⌉	Ceiling Function. If *x* is an integer, return *x*. If *x* is a float, truncate the decimal portion and add 1.
⌊*x*⌋	Floor Function. If *x* is an integer, return *x*. If *x* is a float, truncate the decimal portion.
hash(*x*,*y*)	Calculates hash of x using a hash function, identified by *y*.
Σ(*x*)	Returns $T(Tx \text{ or } Blk) \ T(Blk_\alpha)$, or the time from Blk_α.
σ(x)	Returns $T(Tx \text{ or } Blk) - T(Blk_{R(Tx \text{ or } Blk)})$, or the time from the first block of the TPS interval.
Rwd()	If block, return total block reward without transaction fees. If blockchain, return original block reward.

Experimental Scalability Comparison Model

Controls and Constants

To maintain a controlled test environment that can be reproduced precisely with near-equivalent results, virtual machines (VMs) were rented from a cloud provider, Google Cloud

Platform, where blockchains were run. VMs were chosen due to their modular nature, allowing 1000 machines to be deployed based off of a single template. This simplifies the process of starting up nodes to create a P2P network without having to initialize nodes and peer gossip manually. It also prevented bias as virtualization allows every node to be the same in terms of computational power. With large networks (such as the Bitcoin main network, or "mainnet") having around or upwards of 10000 full nodes, 1000 was a reasonably large network size without incurring a large operating cost. The official Bitcoin Core client was the first blockchain for the model to be tested on, chosen due to Bitcoin being the first blockchain and most widely used in public, making it an ideal choice to gather baseline performance data. To simulate network latency, and due to the "multi-hop" effect of a peer-to-peer network (transactions and blocks may have to cross multiple nodes/machines to get to its destination), the bandwidth was throttled for each node, as the network is not as large as the Bitcoin mainnet, meaning communication would have been inherently faster. Transactions and blocks were queried from randomly selected nodes, from the raw data, median fees were calculated.

Security Dependent Variables

All blockchains have an implicit limit on scalability, denoted $\mathcal{O}(c)$, where c is the average computational power of a single computer in the network, and $\mathcal{O}(c)$ defines the time and storage space requirements of processing c load as less than or equal to c. Considering c is likely not much more than the power of a modern desktop computer, a blockchain can be compared to a central payment mainframe as a microchip credit card to a phone; one is very secure but not meant to compute heavy loads, the other is very fast but subject to a multitude of security vulnerabilities. It is for this reason that constraints and block size and time between block generation are codified to prevent unmanageable load, spam, and to account for inherent latency


on the Internet that is amplified by a P2P network. Although this causes congestion when many transactions are waiting to be added to the blockchain in the unconfirmed transaction memory pool ("mempool"), it is necessary to prevent attackers from gaining control of the network, which would be a greater problem than low throughput. Transactions must be confirmed to be considered secure, or else users can simply replace them with higher fee transactions, known as a double-spend attack. The coded security constants are collectively known as security dependent variables. As this experiment was a model proof for Bitcoin as of now, these variables can be considered constants. For future blockchains to be tested, these security constants will be analyzed on top of the effects of increasing TPS. These come in the form of block difficulty retarget functions, which adjust the difficulty of generating a PoW based on previous block times to cause block generation to tend towards a target time. In this case, Bitcoin had a target time of 10 minutes, and so on. These variables were intrinsic properties of each protocol and were not derived through experimentation. However, all other variables had been derived through experimentation (See *Table 2* for all dependent variables).

Network Bootstrap

A node template was deployed 1000 times simultaneously, yielding 1000 nodes with 1 internal IP address each, sequentially generated. Each node has an identifier from the range of integers 1–1000. From this range (not including the node's identifier in question), 8 were sampled at random for each node, and the node connected to the 8 IP addresses corresponding to the identifier. While not trying to create a full mesh

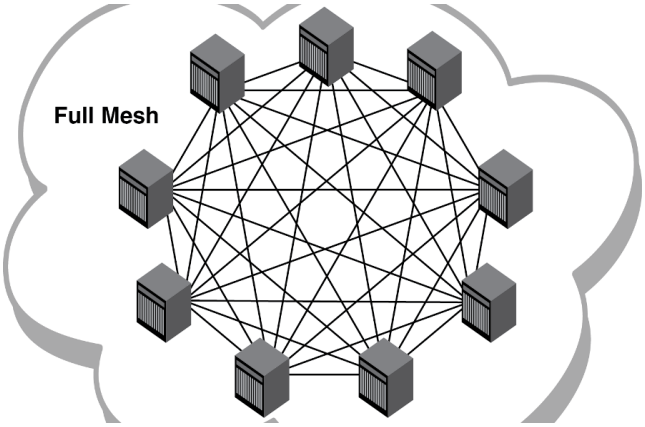

Figure 4. A full mesh network topology, which P2P networks emulate partially.
(Shekbar, 2016)

where all nodes are connected directly (see *Figure 4*), node cliques (sets of adjacent nodes) were large enough that data could move efficiently. In order to prevent bias in the nodes, a separate node not part of the P2P network, known as the command node, sent instructions to other nodes to send transactions and return data.

Model Implementation

Before the experiment was run on a blockchain, every node received a large endowment of pre-generated test bitcoins to their addresses to pay for transactions and associated fees (not to be confused with mainnet bitcoins, as test bitcoins have no real value). Full nodes sent out transactions to random addresses in the set of full nodes in the blockchain (excluding its own address), at a specific rate, long enough so data could be gathered. This was done by selecting nodes randomly to send a specific amount of transactions and then wait. Random nodes are selected, and the number of nodes is inversely proportional to the gap in seconds for their transaction broadcast. This rate was measured for 1 hour, the length of a constant TPS (transactions per second) interval. This was enough to gather the DVs and average them. *PoolSize* is not an average, but a regression function (See *Table 2* for the general form). Intervals started from 1 TPS all the way up to 120 TPS, a total of 5 days of uptime. Most of the scalability DVs were straightforward to derive, except for *PoolSize*. The instantaneous mempool size (number of unconfirmed transactions) had to be measured at various points during a TPS interval, and every time a block is produced, the mempool suddenly decreased. For this reason, a modified sawtooth wave was chosen as the best fit for representing *PoolSize*. The network observed a fundamental property of Bitcoin known as Byzantine Fault Tolerance (BFT), in which a distributed system (such as a P2P network for electronic cash like Bitcoin) can operate


reliably with less than ½ of all nodes compromised or dysfunctional, even with inaccurate information being gossiped.

Table 2. Intermediate and Dependent Variable Definitions

Lowest safe minimum fee during TPS interval in atomic units	$AtomMinFee_r \coloneqq min\big(\Omega\big(\mathrm{Fee}(Tx)\big)\ \forall\, Tx \in B_r\big)$
The maximum block wait time for a *MinFee* transaction	$MaxBlock \coloneqq max\left(\mathcal{O}\big(BT(Tx \mid MinFee)\big)\right)\ \forall\, Tx \in B$
Median transaction fee	$AtomMedFee_r \coloneqq \mathrm{med}(Fee(Tx))\ \forall\, Tx \in B_r$
Maximum time in seconds for a *MinFee*	$MaxTime \equiv MaxBlock \times CT_s$
Average of transaction timestamp vs block timestamp (Unused)	$ConfTime \coloneqq \frac{T(\lceil BH(Tx) \rceil) - T(Tx)}{\lvert B \rvert}\ \forall\, Tx \in B$
Mempool size modified sawtooth wave regression	$PoolSize \equiv poolSize(r) \coloneqq mod(r \times t, \mathrm{CT_s}) + mt - b$
Atomic units per Reward Unit	$RU \coloneqq (Rwd(\mathrm{B}) \times CT_b)$
Cross-chain comparable *MinFee*	$RUMinFee_r \equiv AtomMinFee_r \div RU$
Cross-chain comparable *MedFee*	$RUMedFee_r \equiv AtomMedFee_r \div RU$

Data Extraction

Blockchains, as large data structures, contain well-structured blocks and transactions, each of which has a predictable layout. In this layout, transacted values can be found, collected, and analyzed. Every transaction has an input amount and output amount, the output must be equal to or less than the input value. Usually, it is less than the input, and the remaining bitcoins are sent to miners as fees. Fees are used to discourage spam, with miners prioritizing transactions with higher fees and dropping transactions with low/no fees. For each block in a TPS interval, the raw transactions were collected. For each transaction, outputs were subtracted from inputs, yielding fees. To compare *MedFee* across chains, a system called Reward Units (RUs) was implemented to equate fees (See *Table 2* for the atomic unit to RU conversion). Every 10


seconds during the experiment, all peers were queried for the mempool size, so that *PoolSize* could be derived.

Results

The experiment did not completely accomplish the goal of proving the model due to discrepancies in data output. This was due to a lack of accounting for latency in sequential JSON-RPC requests, magnified over a long period of time. However, most of the code used to command the nodes is properly functioning, only requiring a fix of the query logic and control flow of the Bitcoin model implementation is trivial (See *Future Research* regarding the multi-threading model). Even if there is latency after threading, if all nodes are offset by approximately the same latency, the net latency effect is nullified. Of the results obtained, only *PoolSize* varied over time.

Median Unconfirmed Transaction Pool Size — *PoolSize*

To calculate *PoolSize*, 29960 data points were collected, containing time relative to experiment start and the median size. This is depicted in Figure 5 as a scatter plot. All data points are stored in comma separated values (.csv) files.

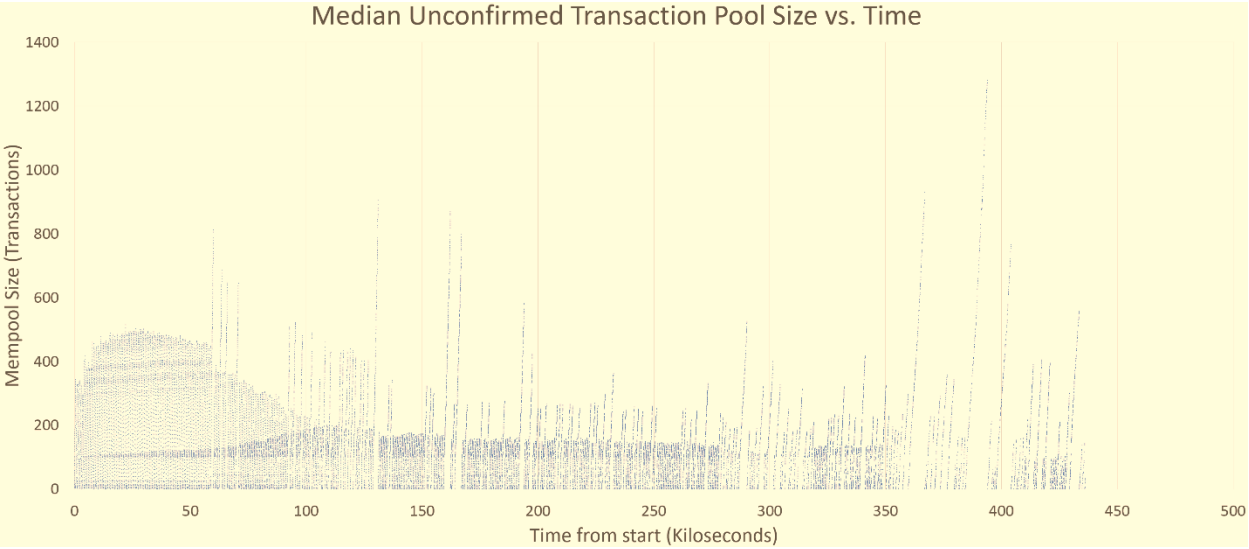

Figure 5. Instantaneous size of the blockchain mempool measured in 10 second intervals as a median of data from all 1000 nodes. The x-axis is in kiloseconds (ks). For reference, 50 ks is approximately 0.58 days.

Minimum Estimated Transaction Fee — *MinFee*

MinFee collection utilized the `estimatesmartfee` method of Bitcoin Core using a maximum block time setting of 1008 blocks (incidentally, half of the difficulty retarget time) and yielded no data as the mempool was never high enough for the fee estimate to be collected. In the future, MinFee will not use an estimate as high as 1008 blocks, as this will never be used in the first place (this is the same as a week for a transaction to clear). At maximum, MinFee will use 90 blocks, which is what is the software's estimate limit according to testing, equivalent to approximately 0.625 days. If the mempool size is high enough and the block setting is set to 90, the MinFee collection function using `estimatesmartfee` will output correct estimates. No function logic needs to be changed, only the control flow.

Median Real Transaction Fee — *MedFee*

MedFee collected the median transaction fee of every TPS period by querying a node for the 6 blocks of the TPS interval and their list of transactions, gathering this list of transactions, and subtracting output bitcoin values from input bitcoin values. The output is the total transacted, the input is references to existing funds, and the difference is the fee paid to the miner. Every difference was collected into a data structure where the median value was found and written to a file. The data structure is then discarded and reconstructed for the next TPS interval. By threading the collection of *MedFee*, latency will no longer be a problem as nodes are selected at random. *MedFee* topped off at a value of 3360 satoshis, or 0.0000336 bitcoins as fees per transaction.

Discussion

Although no *PoolSize* regression could be directly calculated due to the form of the data, several facts can be derived from the resultant plot. Firstly, sequentially querying nodes to send

transactions will result in invalid real TPS, which also leads to timing errors. Secondly, latency of nodes varies throughout time, occasionally latency will drop significantly and/or blocks fail to collate transactions in time due the mining logic being integrated into the command script's main thread. Lastly, so long as average latency of a single query does not change rapidly, if queries are started at strict time intervals apart with a similar latency, the net effect of latency is near zero. Previous studies have formally defined a method of experimentally analyzing deployment blockchain software, with few analytical works concentrated on specific aspects of scalability for novel blockchain technologies such as succinct non-interactive arguments of knowledge (SNARKs), their zero-knowledge variant (zk-SNARKs), ring signatures, Byzantine Agreement (BA) systems, etc. rather than the underlying blockchain protocol itself (Blum *et al.*, 1988) (Rivest *et al.*, 2001) (Lamport *et al.*, 1982) (Croman *et al.*, 2016). Lack of data thereof is problematic as all the above-mentioned technologies rely on blockchains as a substrate for consensus, building scalability on top. It the underlying layer is not scalable; the above layers are impacted in terms of scalability. Decentralization and security may also be lowered when protocol-level amendments (called "hard forks") are made without substantial evidence of their effectiveness (e.g. increasing the block size, reducing difficulty overall can eventually lead to 51% attacks, or total control under a single attacker, which break the BFT paradigm).

Applications

The current model provides insights on how to improve both the model itself, and eventually provide a direction to design better blockchains. Bitcoin can only handle a certain number of transactions at a time, 7 per second as of yet, and requires 10 minutes for a single confirmation, and 1 hour to ensure that a transaction is not reversed. While blockchains have potential to eventually replace existing payment infrastructure and other Internet-based


applications that require consensus in an unstable network, such as stock exchanges. It would be undesirable, as an example, that someone who wants to buy a coffee in a shop that accepts Bitcoin to have to wait an entire hour just so that the seller is confident that a transaction will not be reversed, making the buyer pay a very high fee, sometimes even larger than the size of the purchase itself it extreme cases. Exchanges have to handle thousands of transactions per second, and these transactions must be cleared rapidly. Having both the throughput of a high-performance mainframe and the safety of blockchain consensus is the ultimate goal, which this experimental model provides more insights to.

Future Research

This experiment demonstrates that latency in both P2P and command instance to peer communication must be either eliminated or accounted for in a way that possible error is reduced to zero. Future model implementations built upon the one used for this experiment should contain a parallel, strictly-timed control flow that uses multi-threading to delegate tasks without latency, preventing timing errors from gathering. The behavior of the network demonstrated that, when attempting to query nodes 120 times per second with latency of broadcast being only a few milliseconds, the resultant timing offset that occurs due to the wait for a request to finish before starting another had caused the number of total blocks generated, and the number of transactions to be less than expected, therefore the transaction rates at certain times were not what they were programmed to be.

Conclusion

This experiment was the first step in a series simulated proofs of the blockchain scalability model demonstrated in this paper. The tested model implementation, although sound in appearance, is subject to change as further experimentation reveals more confounding variables in blockchain model implementations, with the model code updated to reflect these

new variables. The model was not proved to be implementable in this trial, and the results clearly establish a next step in the process of developing and testing blockchain scalability; the need for the central commanding node to execute all commands and queries in parallel. By establishing the framework for the first generalized experimental scalability model, future blockchains can be tested and compared against existing blockchains so that research efforts are concentrated on developing blockchains that most satisfy the three desirable properties of the Scalability Trilemma. By eventually solving the trilemma, commercial platforms can be entirely powered by blockchains as a backbone safe from attackers, efficient, and drive the future of all commerce, while absolving unnecessary legal disputes among all parties through reliable, mathematically proved consensus mechanisms.

Works Cited

Back, A. (2002, August 1). Hashcash — A Denial of Service Counter-Measure. Retrieved from http://www.hashcash.org/papers/hashcash.pdf

Bitcoin Avg. Transaction Fee chart. (n.d.). Retrieved August 15, 2018, from https://bitinfocharts.com/comparison/bitcoin-transactionfees.html#1y

Blum, M., Feldman, P., & Micali, S. (1988). Non-interactive zero-knowledge and its applications. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing - STOC 88*. doi:10.1145/62212.62222

Buterin, V. (2015, September 14). On Slow and Fast Block Times. Retrieved August 11, 2018, from https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/

Buterin, V., & Griffith, V. (2017). Casper the Friendly Finality Gadget. *Computing Research Repository (CoRR),* arXiv:1710.09437v2.

Buterin & Griffith detail a new type of PoS consensus mechanism, one that discourages forks and makes finality possible. The new system, called Casper, can only be implemented on blockchains that are Turing-complete.

Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., . . . Wattenhofer, R. (2016). On Scaling Decentralized Blockchains. *Financial Cryptography and Data Security Lecture Notes in Computer Science*,106-125. doi:10.1007/978-3-662-53357-4_8

Hall, T. A., & Keller, S. S. (2014, March 18). *The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS)* (United States, NIST, Information Technology Laboratory). Retrieved from https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/dss2/ecdsa2vs.pdf

Kemp, S. (2018, January 30). Digital in 2018: World's internet users pass the 4 billion mark. Retrieved September 23, 2018, from https://wearesocial.com/blog/2018/01/global-digital-report-2018

King, Sunny, and Nadal, Scott. “PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.” Peercoin, 19 Aug. 2012, https://peercoin.net/assets/paper/peercoin-paper.pdf.

In this article, King & Nadal formally describe the first implementation of a Proof-of-Stake consensus mechanism into a public deployment blockchain.

Koblitz, N. (2001). *Introduction to elliptic curves and modular form*. New York: Springer.

Kwon, J. (2018, June 22). Tendermint Documentation. Retrieved August 11, 2018, from https://media.readthedocs.org/pdf/tendermint/master/tendermint.pdf

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems,4*(3), 382-401. doi:10.1145/357172.357176

Merkle, R. C. (1979). *U.S. Patent No. 4309569(A)*. Washington, DC: U.S. Patent and Trademark Office.

Nakamoto, S. (2008, October 31). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved March, from https://bitcoin.org/bitcoin.pdf/

This article, written by Nakamoto, describes the novel idea of using blockchains for the first time in history. It has not been modified since its original release and was published before Bitcoin's genesis block. While not a formal specification, it describes the core components of Bitcoin's structure.

Percival, C., & Josefsson, S. (2016, August). The scrypt Password-Based Key Derivation Function. Retrieved from https://tools.ietf.org/html/rfc7914

This is IETF RFC 7914, a formal specification of the scrypt PBKDF.

Ray, J. (2018, August 22). On sharding blockchains. Retrieved September 23, 2018, from https://github.com/ethereum/wiki/wiki/Sharding-FAQs#this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it

Rescorla, E. (2018, August). RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3. Retrieved September 23, 2018, from https://tools.ietf.org/html/rfc8446

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. doi:10.21236/ada606588

Rivest, R. L., Shamir, A., & Tauman, Y. (2001). How to Leak a Secret. *Advances in Cryptology — ASIACRYPT 2001 Lecture Notes in Computer Science*,552-565. doi:10.1007/3-540-45682-1_32

Srinivasan, R., (2011, October 01). Mathematics Towards Elliptic Curve Cryptography-by Dr. R.Srinivasan. Retrieved from https://www.slideshare.net/municsaa/mathematics-towards-elliptic-curve-cryptographyby-dr-rsrinivasan

Stark, J. (2018, February 12). Making Sense of Ethereum's Layer 2 Scaling Solutions: State Channels, Plasma, and Truebit. Retrieved August 15, 2018, from https://medium.com/l4-media/making-sense-of-ethereums-layer-2-scaling-solutions-state-channels-plasma-and-truebit-22cb40dcc2f4

Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. *First Monday*,*2*(9). doi:10.5210/fm.v2i9.548

Szilágyi, Péter. *Clique PoA Protocol & Rinkeby PoA Testnet*. 4 Apr. 2017, https://github.com/ethereum/EIPs/issues/225. Accessed 11 Aug. 2018.

Clique is a Proof of Authority consensus engine that allows the securing of test (non-deployment) blockchains using appointed sealers nodes rather than miners or forgers.

United States, NIST, Information Technology Lab. (2015, August). *Secure Hash Standard (FIPS 180-4)*. March 23, 2018, http://dx.doi.org/10.6028/NIST.FIPS.180-4

Vicco, A. (2016, September 8). Code is Law and the Quest for Justice. Retrieved from https://ethereumclassic.github.io/blog/2016-09-09-code-is-law/

Wang, L., & Pustogarov, I. (2017). Towards Better Understanding of Bitcoin Unreachable Peers. *CoRR, Abs/1709.06837*.

Wood, G., & Buterin, V. (2018, June 5). Ethereum: A Secure Decentralized Transaction Ledger. Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf

Ethereum, a decentralized application platform built from a blockchain and virtual machine substrate, is technically defined in this paper. This paper is purely informational and is a

formal specification of the Ethereum protocol, as of the Metropolis vByzantium hard fork consensus rules at mainnet block 4370000. This paper is identifiable by its revision ID, `0xe94ebda`