Design of a Blockchain Protocol Scalability Comparison Apparatus Using Virtual Machines

(Phase II)

Michael Lin, Yorktown High School

Overview

In the past two decades, cryptographic tools and redundancy in the design of computer systems serving clients were implemented gradually over time (Rescorla, 2018), yet still exhibits information disputes due to the inability of computers to identify the canonicity (a.k.a. officiality) of information among disagreeing computers. The "blockchain" is a type of data structure that is both secure and redundant, overcoming the canonicity problem. It finds many uses, most commonly to transact currency in a provably fair method that involves no intermediate bank or payment network. Currently, blockchains, such as Bitcoin, cannot scale to serve as many users compared to a major credit card/payment network without sacrificing security and redundancy. Because current blockchains are, in practice, more secure than centralized payment networks, it is desirable to replace existing schemes with them. To do so needs experimentation and consensus on which blockchains are the most scalable (a.k.a. highest performant), which is the aim of this project. Using virtual machines, a blockchain network can be created and used as a testing environment.

Review of Literature

Digital Authentication

The concept of digital signatures, a core component of transaction security on the blockchain, 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. In addition to this, users can verify data that only the issuer of a message can sign. 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 was 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 in order to stay secure. Because of this, 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 algebraic curve with a special theorem known as the elliptic curve discrete logarithm problem (ECDLP) to replace RSA and similarly designed schemes gained popularity in the past 2 decades. Elliptic curve schemes could do everything that RSA could do with smaller keys (Koblitz, 2001). Simply put, by finding intersection of 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 computationally difficult to find the number of times this operation was performed given only the starting and ending points (see Figure 1), yet solutions involving that private number can be verified. Replacing RSA with elliptic curves meant less space consumption in large databases, where signature data can be reduced by a factor of 10+ times on large datasets.

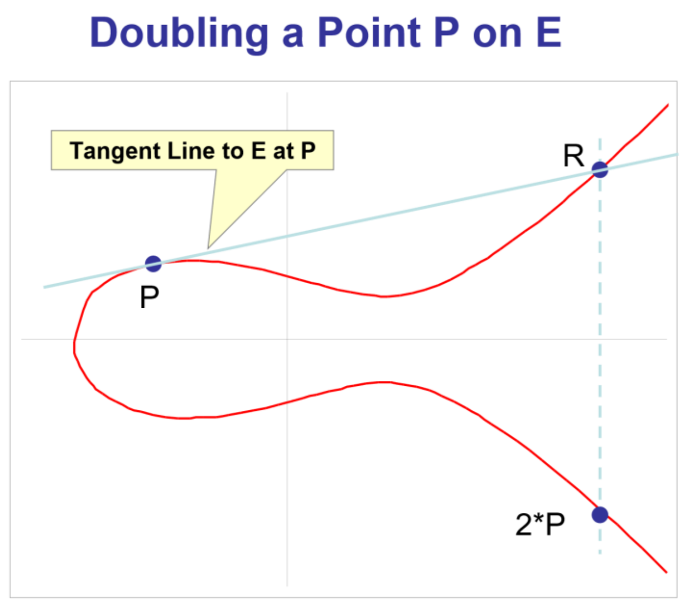

Figure 1. Illustration of a single point multiplication of point P. In practice, many multiplications are used, and the number of multiplications is the private key K, and the public key is the end point. The originating point is referred to as the generator G. (Srinivasan, 2011)

Despite the difference between RSA and EC algorithms, both types employ the use of a cryptographic hash function, or simply “hash functions”. The output of a hash function is known as a hash. Hash functions can be abstractly thought of as a unique labeling algorithm, 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 (e.g. a list of transactions between bank accounts), the naive approach of the set provider would be to sign each element of the set, which would take a long time to sign as well as use more storage space. However, by using a cascade of hashes called a Merkle Tree (see *Figure 2*, below “Block 11”), 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 (Merkle, 1979). In *Figure 2*, to prove *Tx3* is part of a block, only *Hash2, Hash01*, and the signature of *Tx_Root* need to be provided. This is much smaller in size than providing Tx0, Tx1, and Tx2, as hashes are 256-bits long (in the case of Bitcoin). Merkle Trees make blockchains feasible for use on portable devices such as smartphones. Bitcoin and most of its derivative versions use ECDSA and Merkle Trees.

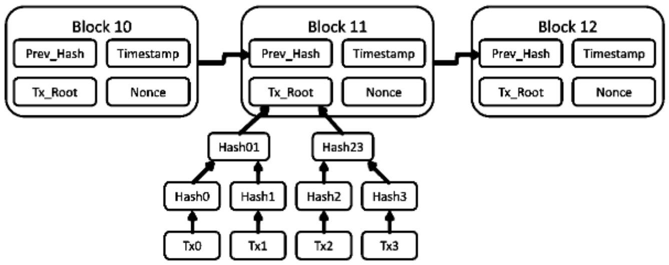

Figure 2. Abstract flow chart showing transactions, a Merkle tree, and blocks, and how they are hashed together
(Blockgeeks, 2017)

Blockchains

Using the Elliptic Curve Digital Signature Algorithm (ECDSA), 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 an open-source and peer-to-peer alternative to traditional banking (Nakamoto, 2008).

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 can perform any operation a regular computer can do, 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.

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, scalability defines the number of transactions or operations that a blockchain can handle before becoming unacceptably congested, decentralization defines how much of the network is controlled by actual users as opposed to centralized nodes, which provide transactions for clients such as mobile phones. Bitcoin is inherently decentralized due to its gossip protocol, but its price and user base grew since it first started. To generate secure blocks that are hard to reverse, one-time computational puzzles are computed by computers known as miners. A block that contains a valid solution to these puzzles is considered to be canonical as a predictable amount of electricity was spent making it. For this effort, miners that successfully find a solution are rewarded with newly minted cryptocurrency. However, miners began to design efficient yet expensive chips to mine faster and consume less electricity. Mining has become more and more difficult, 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 MB 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*).

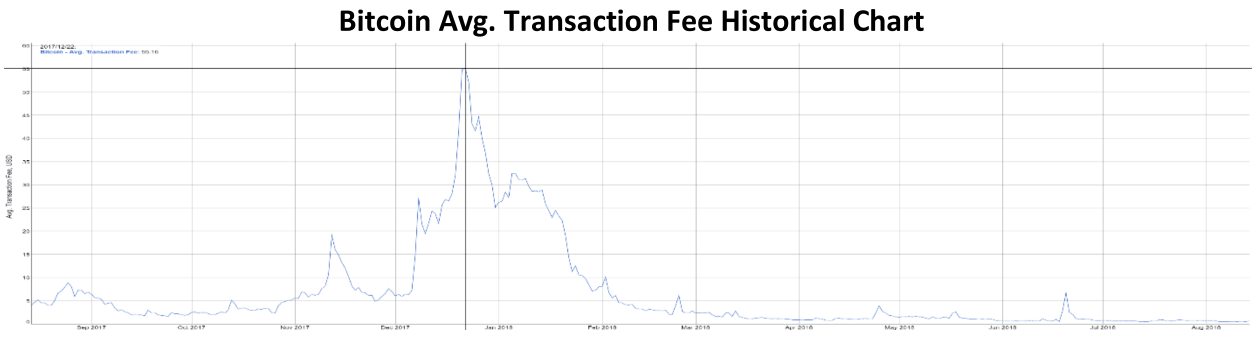

Figure 3. Historically high fees on Bitcoin, peaking at $55.16 USD per transaction. Prices according to exchanges.
(BitInfoCharts, 2018)

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. This is a roundabout method of increasing block size, as miners must 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), and reduces the logarithmic advantage introduced in the last section.

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. Within all of these solutions, a tradeoff within the trilemma would always occur.

In the 2018–2019 *Phase I* study, the student researcher, Michael Lin, under advisory by his project mentor, Dr. Jonah Benton, produced preliminary results on the behavior of a serial implementation of the model. Phase II of 2019–2020 modifies the model implementation to run processes with precise timing and moves the gossip regulator to the session layer, with the goal of improving upon the transaction automation aspect of `bitcoin-scale-test` (version 1).

Problem Statement

- The *Phase I* study of this project utilized serial execution for both transaction gossip and data collection and has yet to produce a comparable data for differing blockchains. Timing must be precise for data to be comparable and objective.

Objective

- To improve upon the work of Phase I with more efficient transaction automation, implement and test `bitcoin-scale-test-v2` on a 1000 node regression test network, using multiple command nodes at fractional TPS and `throttle-proxy` as a gossip regulator, and collect the following variables:
    1. *MinFee* — Minimum fee estimate for a 6 block maximum confirmation time
    2. *MedFee* — Median fee per transaction within a TPS period
    3. *MemPool* — Instant memory pool size, measured in transactions.

Hypothesis

- After running `bitcoin-scale-test-v2`, the following independent variables will not exhibit a R^2 coefficient greater than 95% and a sum of absolute error (SAE) of less than 100 nano-reward units (nRU) for their respective regression curves:
    1. *MinFee* — Polynomial curves of up to degree 10, or exponential curves of up to 10 significant figures, or a linear curve
    2. *MedFee* — Polynomial curves of up to degree 10, or exponential curves of up to 10 significant figures, or a linear curve

3. *MemPool* — A linear sawtooth curve function of up to 10 significant figures. The curve has 4 parameters, one of them fixed to the block time of 600 seconds (p). The curve is defined by $MemPool(t) = n \cdot mod(t, p) + mt + b$

Methodology

The experiment was conducted in the summer before and during the post-Labor Day school year of 2019-2020 as Phase II of the project. I was responsible for the entirety of scripting the conjuring, bootstrap, and automation scripts, and my mentor assisted me with utilizing `SciPy` and `pandas` for data analysis and presentation. Bitcoin Core 0.18.1 is the version used. `bitcoin-scale-test-v2` is designed to be run on a Python 3.8 interpreter. All modules shall be the latest minor release of the major release it was as of September 18th 2019, or the latest version with compatible syntax with it, whichever is the earlier one, for experiment replication. All script filenames are elements of the `bitcoin-scale-test-v2` repository, master branch. Object names shall be globals of the repository or included in the Python Standard Library. `throttle-proxy` is a SOCKS5 proxy server written in NodeJS that enables up/down bandwidth throttling and packet delaying (artificial latency).

Node Conjuring

The development of the new codebase utilized local virtual machine testing. To do so, a fresh Arch Linux virtual machine was created. A simple shell script, `multistart.sh`, was written to create all node data directories, initialize `throttle-proxy` listeners in `screen` sockets, and initialize `bitcoind` instances in separate `screen` sockets (connected to the proxies). Each node had its own numerical identifier, starting at 1. This step, known as node conjuring, creates the nodes that make up the network, so that those nodes can be connected through node bootstrap.

Node Bootstrap

After all nodes had started up, they were bootstrapped with private keys that correspond to vanity addresses, and connected to 8 other peers selected at random. These steps were performed with `bootstrap.py`.

Endowment

To send bitcoins, all nodes required an endowment of Bitcoins to send transactions. Node 1 generated 1322 blocks to endow itself with bitcoins, which are needed in order to send transactions. It then delegated these bitcoins to each of the nodes in groups of 10 vector outputs per transaction. After all transactions were made such that each address had enough bitcoins to operate, the memory pool was mined into the 1323rd block. Testing/experimentation then commenced.

Global Parameters

The main script file, `bitcoin_auto.py`, first imported global variables from `settings.py`, then executed `main()` when called using a Python 3.7 interpreter. The main loop executed under the condition of the current UNIX epoch time being under a specific timestamp, `timeout`, returned by `settings.py`. Time is measured in truncated output from the builtin `time()` function, cast to an `int`. The floor of time in seconds relative to `genesis` (when `bitcoin_auto.py` was called) divided by 3600, plus the `starttps` variable (set at 1 for the main experiment), was equivalent to the current TPS rate.

Transaction Automation

Similar to the code of `bitcoin-scale-test`, the command nodes executed transactions in parallel. However, 30 command nodes were deployed, each handling 1/30th of the exerted TPS. For this reason, the maximum TPS each node had to handle was 1 TPS. Each was timed to offset their transaction calls precisely in a sequence. For example, at 2 TPS, the first command node will execute, 0.5 seconds later the second node will execute, each running every 15 seconds and spaced 0.5 seconds apart.

Mining Automation

A separate command node was conjured for the purposes of mining and data collection. Because of the sparsity of these tasks in comparison the transaction automation, it was logical to have this node handle both operations. Every 10 minutes, `mine()`was executed.

Data Collection

After `mine()` was run from `main()`, `tocollect` was passed to `collect()` as the only parameter. When `tocollect` was 0, nothing was collected. When `tocollect` was 1, *MinFee* and *MemPool* were collected. When `tocollect` was 2, *MinFee*, *MedFee*, and *MemPool* were collected.

All functions called by `collect()` were defined in `data_collection.py`. `minFee()` collects the truncated average of the JSON-RPC method `estimatesmartfee` with a block parameter of 6 in conservative mode. `medFee()` collects and serializes all transactions included in the past 6 blocks (a single TPS period), then calculates the fees of each transaction by the difference of the sum of vector inputs to sum of vector outputs. The median of this fee list is returned. Both of these functions return fees in satoshis, which are elementary (also known as atomic) units of currency equivalent of 10^{-8} bitcoins. `memPool()` returned the median of the memory pool size, measured in transactions.

Data Analysis

All data was exported from the local storage of the master node. This data was stored in comma-separated-value (CSV) files. The syntax is very basic, each line is a data point which exists as a sequence of time-scalar pairs. There existed a CSV file for each independent variable, and these files are parsed into Python objects. Four functions exist that model the curves to fit the regression: A polynomial regression of standard form (degree 10), an exponential regression, a linear regression, the modified sawtooth curve defined using the modulo operator. For every TPS period, an excerpt from the MemPool dataset for that TPS was analyzed.

Results/Analysis

Network Behavior

It is known that the project was bootstrapped evenly, and the network had no discernible centralization. This can be observed in the visual of *Figure 4*. The 30 command nodes executed transaction automation precisely.

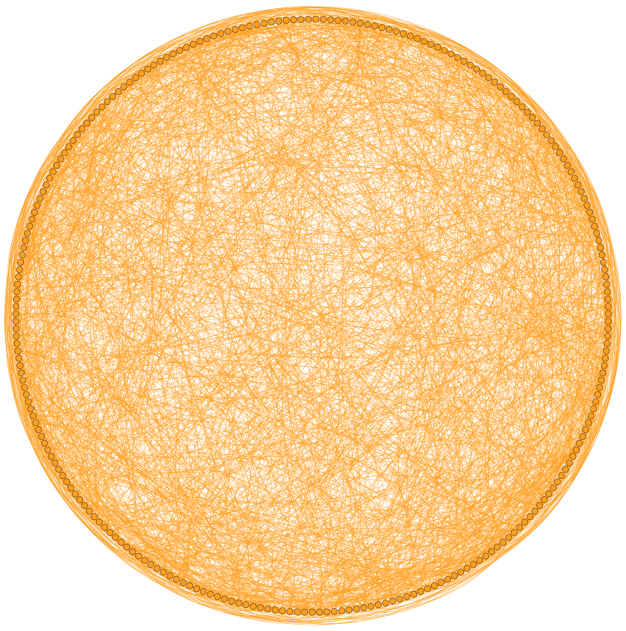

Figure 4. Visualization of the network structure. Every edge dot represents a network node. Every line represents a latent connection between nodes. As visualized, the experiment network is an evenly connected mesh with no biases.

MinFee & MedFee

The data for *MinFee* found no significant correlation for polynomial, exponential, or linear regressions. This is a result of the apparent TPS from the network being lower than the actual commanded TPS. This is true of all TPS periods. This is also true of *MedFee*. Both variables had not had any significant changes over time/TPS. With a precise apparent TPS, this relationship is expected to change (though not necessarily significantly).

MemPool

It had been found that MemPool had a strong r^2 for 30% of the TPS periods (a total of 9), with inconsistency in certain periods observed to be caused by incorrect mining automation. This suggests that the independent mining/data collection command node has timing issues. However, with the 9 periods that had significance, the value of *n* had found a strong linear correlation of r^2=93.9%, suggesting that network automation had occurred correctly. In the case of this regression curve, n represents the average of the derivatives of apparent TPS between every block within a TPS period. Because it describes apparent TPS, the values can be compared against the commanded TPS. See the following: *Figures 5, 6 & 7*.

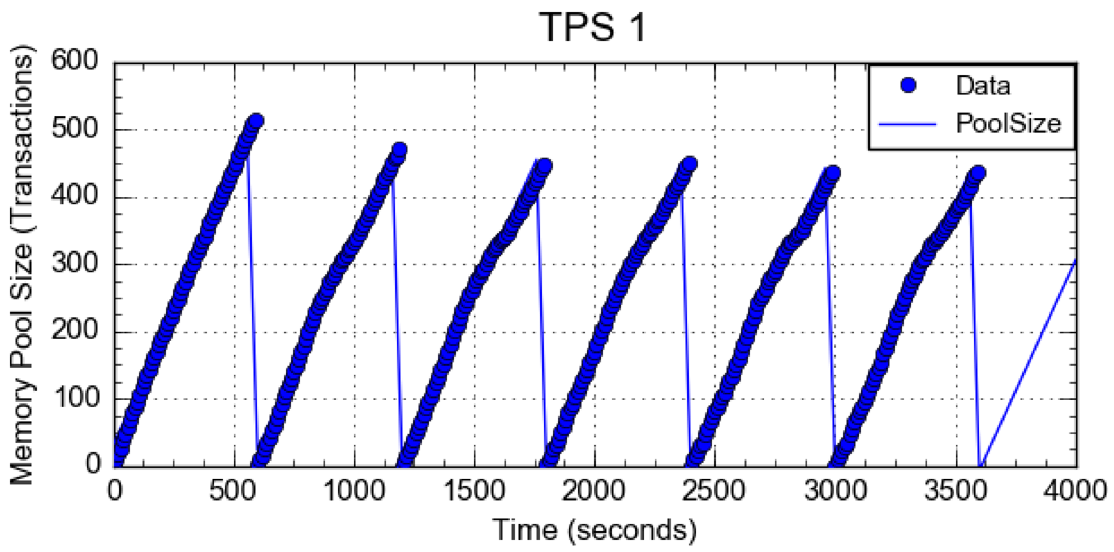

Figure 5. Observation of the first TPS period, along with the PoolSize regression with fitted parameters. A definite relationship can be seen.

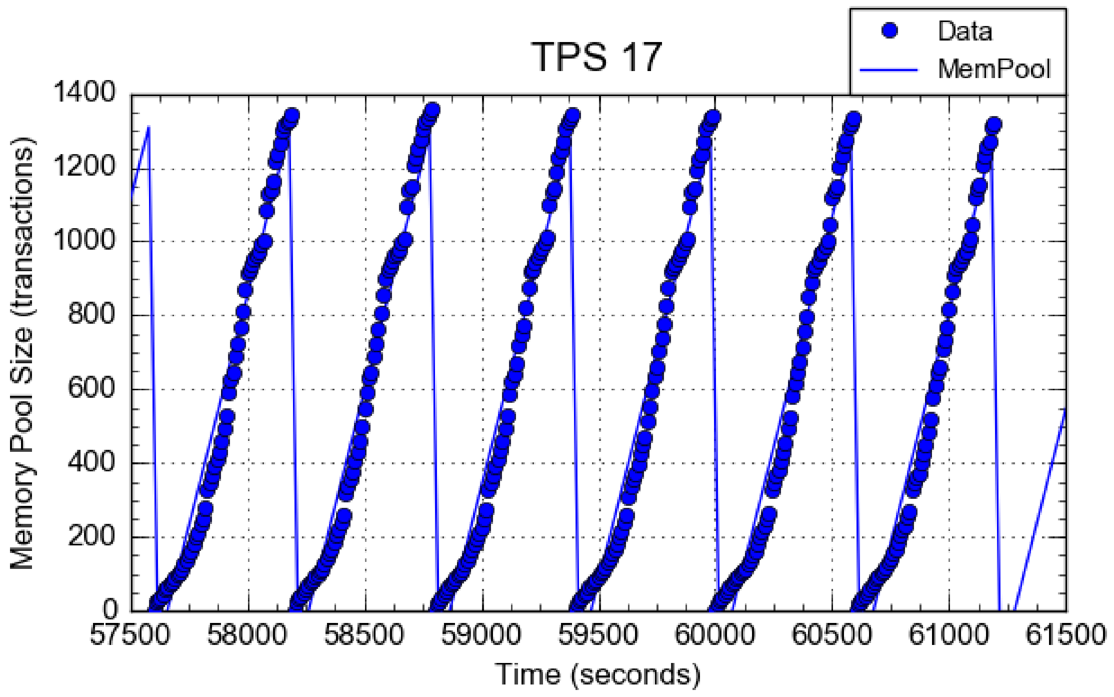

Figure 6. Observation of the 17th TPS period, with a steeper fit than the first period. The same definite relationship can be seen, albeit steeper. Note that if apparent TPS was equal to commanded TPS, the peak-to-peak values of this curve would be approximately 10200, instead of 1400.

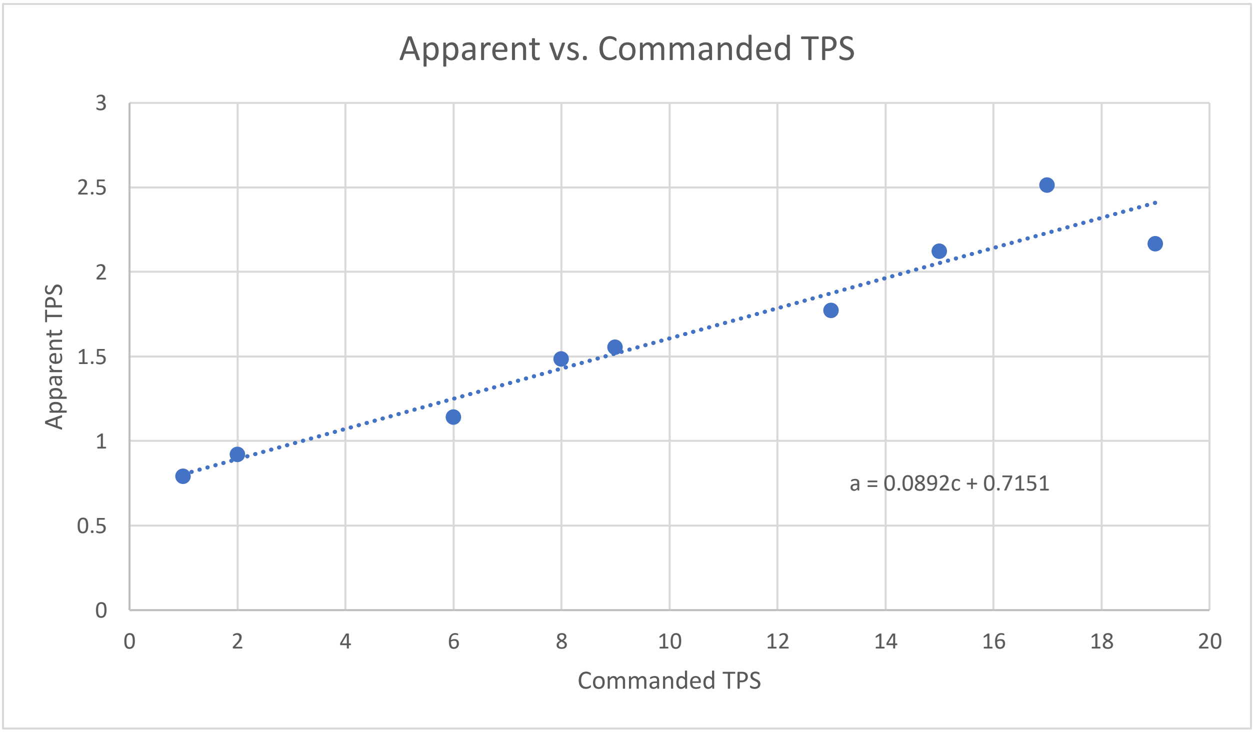

Figure 7. Relationship between the network-observed ("apparent") TPS, which affects fee estimation operations, versus the actual user load, or "commanded" TPS.

Application/Future Research

It can be stated that the relationship of commanded TPS versus apparent TPS is therefore c=11.210762a, without accounting for initial values. This can be used to precisely designate an apparent TPS, and observe the network's reaction to integer value increases of apparent TPS.

The cause the disparity between apparent and commanded TPS is likely a cause of Byzantine conditions of the network. As mentioned, all connections between network nodes is latent, resulting in transactions and blocks propagating gradually across the network, and imperfect information about the network. The linear correlation between actual user load (which is not quantifiable in real, public blockchains) and observable load from network statistics is novel information, and can be used by developers in more precise testing of their blockchains. The use of private networks introduced by this experiment are more stable than deployment test networks (also known as "testnets"), due to developers having full knowledge of which transactions, and how quickly such transactions are being introduced to the network.

Future research should attempt to answer the questions of: "What is the relationship between bandwidth constriction and TPS disparity?", "How well do fee variables correlate with precisely controlled apparent TPS?", "What are the effects of a higher O(c), also known as average network computation power, or a more powerful set of command nodes?" With a finished model developed by answering these questions, cross-chain comparisons can be made with separate experiments, using the Reward Unit model the equates atomic cryptocurrencies of different chains.

Conclusion

This experiment was the second step in a series simulated proofs of the blockchain scalability model demonstrated in this paper. The tested model implementation has found unexpected, yet novel results that demonstrates its viability with further improvements. The model was shown to be more precise than the previous study conducted in 2018, and yields the way for future experimentation and use by other researchers. 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.

References

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