Virtual-Time-Accelerated Emulation for Blockchain Network and Application Evaluation

Xiaoliang Wu
Illinois Institute of Technology
Chicago, Illinois
xwu64@hawk.iit.edu

Jiaqi Yan
Illinois Institute of Technology
Chicago, Illinois
jyan31@hawk.iit.edu

Dong Jin
Illinois Institute of Technology
Chicago, Illinois
dong.jin@iit.edu

ABSTRACT

Blockchain technologies are in the ascendant of transforming the ways we manage contracts, make transactions, and manifest ownership of property. The trend calls for a realistic testing and evaluation platform for blockchain applications and systems. We present Minichain, a container-based emulator that allows testing proof-of-work-based blockchains on a commodity computer. Minichain contains a realistic and configurable network environment, which is missing in today's blockchain testbeds. This unique feature enables us to evaluate the impact of network events (e.g., cyber-attacks) and conditions (e.g., congested or failed links) on blockchain applications. Meanwhile, Minichain allows the direct execution of unmodified application code in the containers for fidelity, and utilizes the virtual time technique to speed up experiments and improve the system scale that one can accurately emulate. In particular, we mathematically analyze the convergence of the proof-of-work-based consensus algorithm to show the effectiveness of virtual time. We evaluate the performance of Minichain across both network layer and application layer, and demonstrate its usability by emulating a selfish mining attack initiated from the network layer.

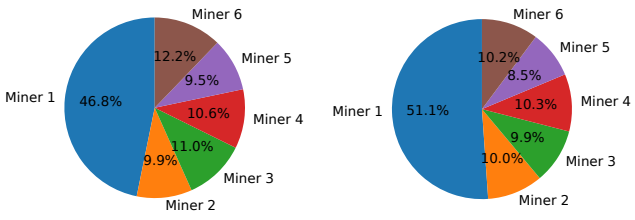
ACM Reference Format:
Xiaoliang Wu, Jiaqi Yan, and Dong Jin. 2019. Virtual-Time-Accelerated Emulation for Blockchain Network and Application Evaluation. In *SIGSIM Principles of Advanced Discrete Simulation (SIGSIM-PADS '19), June 3–5, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3316480.3322889

1 INTRODUCTION

Blockchain technology promises an open, distributed, and secure digital ledger for recording transactions and contracts. Today, new ideas based on blockchain technology are springing up all over financial, legal, and political systems [1, 3, 4]. Specific blockchain-based projects [6, 7, 22, 23] draw tremendous public attention not only because of the openness and decentralization offered by blockchains, but also the innovative application design and the claimed competitive system performance like high transaction throughput, short read latency, and high amount of coins. However, as a blockchain is a complex peer-to-peer distributed system, a convincing experimental analysis is essential to support the theoretically claimed performance advantages before the actual deployment, as people certainly prefer witnessing the healthy operation of a working blockchain to a lengthy and dry whitepaper.

Several blockchain testbeds were proposed by the research community to meet this emerging demand [5, 24]. The testbeds are often established on private clusters, and the physical machines running blockchain applications are directly connected to form small-scale testing blockchains. The testbeds either do not include the underlying communication network or only offer a static network environment with no interface to configure. However, blockchain practitioners have realized the critical impact of network environment on blockchain applications [29]. Furthermore, blockchains are increasingly used to boost cyber-security across industries. Evaluating the effectiveness of those new applications against cyber-attacks also calls for a realistic network layer in a blockchain testbed. Let us demonstrate the importance of considering the network environment in a blockchain with the following motivating example.

We set up the following scenario with six miners to explore a known vulnerability of the Ethereum system [29]. All miners keep mining with a one-second average block generation interval until the 1000-th block is generated. Figure 1 depicts the distribution of blocks under different network latency conditions. In the high-latency network (400 ms link delay), miner 1 publishes more than 50% of blocks (51.1% as shown in Figure 1b). In other words, miner 1 has successfully launched the selfish mining attack [9, 22] so that it has the potential to break down nearly all the security properties that a blockchain-based ledger promises. For example, miner 1 now can prevent new transactions from happening, reverse early-signed contracts, or even double-spend Ethereum coins. However, the same issue is not observed in the low-latency network scenario (10 ms link delay) as shown in Figure 1a, where miner 1 possesses <50% of the blocks.



(a) 10 ms Link Delays (b) 400 ms Link Delays

Figure 1: Comparison of Distributions of Blocks with Different Network Delay Settings.

SIGSIM-PADS '19, June 3–5, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6723-3/19/06…$15.00
https://doi.org/10.1145/3316480.3322889

The difference motivates us to design a new blockchain testbed that contains a realistic network layer and allows easy configuration of the network characteristics, such as link delay, bandwidth, and packet loss rate. Integration of a working blockchain into a network testbed offers a solution. Among the various types of network testbeds, physical network testbeds typically do not offer desired controllability over the network, and the scale of the network relies on the available hardware; network simulation testbeds, on the other hand, do not offer the desired fidelity due to model abstraction and simplification. In this paper, we develop a container-based network emulation testbed, Minichain, as it provides a good balance between fidelity (by executing unmodified blockchain applications in containers) and scalability (by emulating a network of dozens of blockchain nodes on a commodity server). Minichain also offers users the fine-grained control over the network settings as well as resource management (e.g., CPU slice) among the blockchain peers.

However, integrating blockchains onto a network emulation has unique challenges. Blockchain applications require resource-intensive mining processes to add blocks cryptographically, and the consensus models are often based on computationally expensive algorithms (e.g., proof-of-work). As a result, emulating a blockchain may overwhelm the resources of the underlying physical machine, and hence limit the scale of the system to emulate as well as slow down the execution speed. A key challenge is how to speed up the blockchain emulation without losing fidelity.

Solutions to address this challenge are (1) increasing system resources and/or (2) reducing the required computation. The former includes upgrading the hardware or designing a distributed version of the testbed. However, being able to evaluate blockchains on a single commodity server is a pretty handy feature that we intend to keep. Therefore, in this work, we take the latter approach to reducing computational cost needed to emulate a blockchain while preserving the desired fidelity. Our solution is centered around the application of virtual time in network emulation. In the literature, virtual time is used for a different purpose in network emulation [12, 15, 18, 27]. In particular, when the physical system lacks resources to run an emulation experiment, using virtual time can extend the execution time for generating accurate experimental results. In this work, we explore the virtual time differently with the goal of accelerating the emulation experiment. We notice that miners race to solve computation-intensive mathematical puzzles in traditional proof-of-work based blockchains. To maintain a stable average block interval (e.g., 10 min for Bitcoin) as miners join and leave the chain, the system dynamically adjusts the difficulty in the proof-of-work algorithm. The difficulty has a direct impact on the total amount of computation needed. Therefore, we can leverage virtual time to quickly advance the system state to the future. By preserving the same block interval, the difficulty is reduced, and thus the computational cost is also reduced, which further accelerates the emulation experiment. To fully explore virtual time in Minichain, we first conduct a mathematical analysis of blockchain's convergence with and without virtual time; we then integrate virtual time system to Minichain; finally, we demonstrate that the virtual time is able to speed up the experiment with little fidelity loss in both benchmarking experiments and a case study.

Our contributions of this work are summarized as follows.

- We develop Minichain, a container-based blockchain emulator, for the first time, to allow users to test and evaluate a blockchain system on a commodity machine with a realistic and configurable network environment.
- To speed up the emulation experiment without losing fidelity, we analyze and realize the virtual time support in Minichain. In particular, we mathematically analyze the convergence of the proof-of-work-based consensus algorithm, which enables Minichain to achieve higher efficiency with virtual time.
- We evaluate the system performance of Minichain including the blockchain applications layer, the network layer and virtual time. We also demonstrate how to use Minichain for security analysis on an Ethereum blockchain with a case study.

In the remainder of the paper, we introduce the background knowledge about blockchain in Section 2, and mathematically prove the convergence of proof-of-work difficulty in Section 3 to offer the theoretical support of using virtual time in Minichain. We then elaborate on the challenges a blockchain emulator faces with the goal of preserving the fidelity of selected performance metrics in Section 4 and how we address those challenges in Section 5. We develop a working prototype system, Minichain, and report performance evaluation in Section 6. We further demonstrate the usability of Minichain with a selfish mining attack case study in Section 7. We discuss related work in Section 8 and conclude the paper with future work in Section 9.

2 BACKGROUND OF PUBLIC BLOCKCHAIN

2.1 Proof of Work Consensus in Blockchain

A blockchain is an incorruptible, decentralized, and distributed ledger of programmable and crypotographically-signed transactions that record virtually anything of value. It is also literally a chain of unique data structures (blocks). A block along with a group of stored transactions is resistant to modification as it cryptographically links to the previous block with proper validation. Peers in the blockchain (called miners) are rewarded with coins of financial value (e.g., Bitcoin) when they successfully append a new legitimate block to the main chain. To prevent miners from dictating or confusing the main chain (i.e., everyone has the permission to read and write), a trustless public blockchain system relies on a consensus algorithm to determine who to publish the next block.

Today, most existing public blockchains adopt proof of work (PoW) to select the winner, while promising alternatives exist, such as proof of stake (PoS) and delegated proof of stake (DPoS). PoW works by allowing every untrustworthy party to solve a problem to prove their computational capacity fairly. The selected problem is mathematically hard to solve but easy to verify. For example, the Bitcoin blockchain uses the hashcash where miners exhaustively search a nonce whose hashed value is less than a target value.

PoW applied in a blockchain can be further illustrated by how Alice and Bob commit a transaction:

(1) Bob shares the public key of his account with Alice.
(2) Alice bridges the transaction from Alice's public key to Bob's public key and signs the transaction with her private key.
(3) Alice broadcasts the signed transaction to the entire network.

(4) Miners verify the transaction and seal the transaction into one block.
(5) Miners compete by solving a hard problem until one miner discovers the valid answer.
(6) The winning miner broadcasts the valid block to the entire network.
(7) Bob and Alice confirm the transaction when they receive the block that contains their transaction.

2.2 Key Emulation Performance Indicators

One problem with PoW is how to set an appropriate difficulty. Solving the problem using proof-of-work is meaningless to the transaction made between Alice and Bob other than facilitating the agreement on which miner publishes the transaction. On one hand, making a problem too hard means less computational resources can be allocated to verify and store transactions. On the other hand, the next block is likely to be found by multiple miners if the difficulty remains easy, making consensus hard and slow to emerge. A PoW algorithm dynamically adjusts its difficulty based on the system's available computational resources. Since blockchain applications and users always demand their submitted transactions to be confirmed at a stable and reasonably fast speed, such difficulty adjustment keeps the interval of block generation at a desired stable value as miners join and leave. In this way, a blockchain balances the interests of users of the digital ledger (e.g., Alice and Bob) against the consensus of the flow-in-and-out computational capacity (e.g., miners).

The transaction-making example mentioned above also demonstrates the key emulation performance indicators (KEPIs) that a blockchain testing system ought to focus. We consider the block interval (ideally a stable and reasonably fast value) as a KEPI but not the PoW's difficulty, as it signifies the completion of the first step of transaction confirmation in a block. To prevent double spending, a transaction confirmation is not complete until the block holding the transaction has depth with a certain number of blocks. Therefore, the transaction throughput (i.e., the number of completed transactions per time unit) is another KEPI. Finally, the end-to-end communication between Alice and Bob and the block broadcast by miners rely on the reliable underlying networking infrastructure. Traditional performance metrics like network throughput and end-to-end latency should be considered as KEPIs in the context of this paper.

In summary, we consider block interval, transaction throughput, network throughput and latency as the KEPIs, and the design goal of our blockchain testing system focuses on the fidelity of those KEPIs. We exclude the difficulty of the PoW, as it does not affect the KEPIs in a PoW-based blockchain as long as the difficulty reaches its convergence state.

3 CONVERGENCE OF POW DIFFICULTY

Performing proof-of-work plays an indispensable role in achieving consensus in a blockchain system. However, PoW is designed in a way such that its difficulty is independent of the block interval and the transaction throughput when a blockchain reaches a steady state. In this section, we use a specific blockchain, Ethereum, to demonstrate (1) how a blockchain converges to a steady state, (2) what a converged blockchain looks like, and (3) the mathematical proof that difficulty is irrelevant to KEPIs during the steady state. Ethereum generates blocks as follows:

(1) All miners start off mining a new block b_i by solving a problem with difficulty D_i, in which miners need to find a new piece of data that hashes to a value less than the pre-defined target, which takes D_i expected hash operations.
(2) When a new candidate block is found and published by a certain miner, other miners stop mining to verify block b_i. Once verified, b_i is appended to the main chain and miners repeat the mining step. We assume b_i is always valid for simplicity.

Block interval T_i (i.e., the duration of the block generation cycle) is one KEPI of interest in a blockchain. We define $T(D_i)$ with respect to the current difficulty D_i of the PoW as $T(D_i) = M(D_i) + V(D_i)$ where $M(D_i)$ is the mining time, $V(D_i)$ the verification time. Since $V(D_i) \ll M(D_i)$ in most real-world applications, we ignore $V(D_i)$ in the hereafter analysis. All the assumptions made in both this section and Section 5 are summarized in Table 1.

Difficulty D_i is defined to be the **expected** number of hashing operations to discover the first valid i-th block. PoW dynamically adjusts the difficulty D_i. Particularly, Ethereum's dynamic adjustment algorithm [8] is expressed as:

$$D_{i+1} = D_i + \left\lfloor \frac{D_i}{2048} \right\rfloor \times \max(1 - \left\lfloor \frac{T_i}{10} \right\rfloor, -99) + \lfloor 2^{\lfloor i/100,000 \rfloor - 2} \rfloor \tag{1}$$

The hash function used by PoW generally leaves miners no strategy other than enumerating exhaustively. On one hand, since valid results are not unique and are distributed evenly in the hash function's input and output spaces, it is not guaranteed after D_i hash operations a miner can found a valid block; on the other hand, a miner may also report a valid block before D_i hash operations. The second component in Equation 1 adjusts the previous difficulty D_i depending on how much time is spent on finding the i-th block (T_i). The last component in Equation 1, $bn(i) = \lfloor 2^{\lfloor i/100,000 \rfloor - 2} \rfloor$, is mainly for Ethereum's future support of proof of stake design. In normal emulation, block number i seldom goes beyond 200,000, which makes $bn(i) \leq 1$. Therefore, we ignore $bn(i)$ hereafter from Ethereum's PoW difficulty analysis.

Assume all the miners are mining the blocks at an aggregated hash rate HR (e.g., number of hash operations performed per second), the average block interval (ignoring the time spent on validation) can be calculated as $T_i = \frac{D_i}{HR}$ which renders Equation 1 to

$$D_{i+1} = D_i + \left\lfloor \frac{D_i}{2048} \right\rfloor \times \max(1 - \left\lfloor \frac{D_i/HR}{10} \right\rfloor, -99) \tag{2}$$

We eliminate the *max* function by splitting the recursive formula 2 into two cases

$$D_{i+1} = \begin{cases} D_i + \left\lfloor \frac{D_i}{2048} \right\rfloor \times (1 - \left\lfloor \frac{D_i/HR}{10} \right\rfloor), & 0 \leq D_i < 1000HR \\ D_i - 99\left\lfloor \frac{D_i}{2048} \right\rfloor, & D_i \geq 1000HR \end{cases} \tag{3}$$

3.1 Convergence without Floor Function

Without loss of generality, we replace integer divisions with normal divisions to make the proof concise. We define $\alpha_1 = (1 + \frac{1}{2048})$,

$\beta = -\frac{1}{20,480HR}$, $\alpha_2 = (1 - \frac{99}{2048})$, and $HP_E = 1000HR$ so that Equation 3 is simplified to

$$D_{i+1} = \begin{cases} \beta D_i^2 + \alpha_1 D_i, & D_i < HP_E \\ \alpha_2 D_i, & D_i \geq HP_E \end{cases} \quad (4)$$

The dividing line $D_i = HP_E = 1000HR$ in Equation 4 is defined as follows.

Definition 3.1. Explicit Hash Power, denoted as HP_E, is the total number of hash operations a blockchain system performs for 1000 seconds of PoW with a constant hash rate HR.

If the computational power consumed by the current block is higher than HP_E, Ethereum exponentially decreases the next block's difficulty until reaching a converging point. Otherwise, the difficulty is adjusted quadratically. We divide the difficulty adjustment into two cases, and conquer each case in the following sub-sections.

3.1.1 Convergence of Case 1. Let us consider the case that the current blockchain is under the explicit hash power $D_i < HP_E = 1000HR$,

$$D_{i+1} = \beta D_i^2 + \alpha_1 D_i \quad (5)$$

Definition 3.2. Implicit Hash Power, denoted as HP_I, is the total number of hash operations a blockchain system performs for 10 seconds of PoW with a constant hash rate HR.

Theorem 3.3 (Convergence Below HP_E). *The difficulty converges to the implicit hash power HP_I (see Equation 5). More concretely,*

- *If $HP_I \leq D_i < HP_E$, D_i monotonically decreases to HP_I;*
- *If $0 < D_i < HP_I$, D_i monotonically increases to HP_I.*

Proof. The proof is based on the following lemma about monotone sequence's convergence.

Lemma 3.4. *If a sequence of real numbers is increasing and bounded above, then its supremum is the limit.*

Lemma 3.5. *If a sequence of real numbers is decreasing and bounded below, then its infimum is the limit.*

Let $f(D_i) = \beta D_i^2 + \alpha_1 D_i$, where $D_i \leq HP_E$. With $D_i \leq HP_E < -\frac{\alpha_1}{2\beta} = 10245HR$, we take f's derivative as

$$f(D_i)' = 2\beta D_i + \alpha_1 > 0$$

Therefore, $\{D_i\}$ is a monotonic sequence. Note that we still need to determine if $\{D_i\}$ is monotonically increasing or decreasing. Without loss of generality, we introduce an arbitrary constant $0 < c \leq 1000$, and assume $D_i = c \times HR$. Using the recursive relation defined in (5), D_{i+1} can be expressed as

$$\begin{aligned} D_{i+1} &= \beta(c \times HR)^2 + \alpha_1(c \times HR) \\ &= \beta c^2 HR^2 + \alpha_1 cHR \\ &= c \times HR \times (1 + \frac{1}{2048} - \frac{c}{20480}) \end{aligned}$$

In other words, we have

$$\begin{cases} D_{i+1} \geq D_i, & 0 < c < 10, \\ D_{i+1} < D_i, & 10 \leq c < 1000, \end{cases} \quad (6)$$

which indicates the convergence of sequence $\{D_i\}$:

- when $0 < D_i < HP_I = 10HR$, $\{D_i\}$ monotonically increases. By Lemma 3.4, $\{D_i\}$ converges to HP_I.
- when $HP_I = 10HR \leq D_i < HP_E$, $\{D_i\}$ monotonically decreases. By Lemma 3.5, $\{D_i\}$ converges to HP_I.

Therefore, the difficulty converges to the implicit hash power HP_I, since D_i converges to the same limit $10HR$ in the both cases. □

3.1.2 Convergence of Case 2. $\{D_i\}$ in the case 2 is a geometric sequence with a common ratio $\alpha_2 < 1$. Instead of converging to zero, D_i decreases exponentially to HP_E. Then, D_i falls into the case described in Equation 5 and continues decreasing monotonically to HP_I. In summary, if we replace integer divisions with normal divisions, the following theorem holds true (visually illustrated by Figure 2).

Theorem 3.6 (Convergence of Simplified Ethereum's PoW Difficulty). *With the adjustment algorithm defined as*

$$D_{i+1} = D_i + \frac{D_i}{2048} \times \max(1 - \frac{D_i}{10HR}, -99) \quad (7)$$

$D_\infty \equiv \lim_{i\to\infty} D_i = HP_I$ regardless of the initial value.

3.2 Convergence with Floor Function

Although the integer division does not affect the convergence of Ethereum's difficulty, it affects how the difficulty converges. Assume $HR \geq 2^{20}$ (i.e., at least 1M hash per second), we describe the convergence with the following theorem.

Theorem 3.7 (Convergence of Ethereum's PoW Difficulty). *With the adjustment algorithm defined as*

$$D_{i+1} = D_i + \left\lfloor \frac{D_i}{2048} \right\rfloor \times \max(1 - \left\lfloor \frac{D_i/HR}{10} \right\rfloor, -99) \quad (8)$$

sequence $\{D_i\}$ converges in three different ways depending on the initial difficulty values. More specifically, $\forall i \in \mathbf{Z}^+$:

- *Monotonically-decrease convergence case: $\forall i > 0, D_{i+1} \leq D_i$ and $\lim_{i\to\infty} D_i = 2HP_I$ when $2HP_I \leq D_0$*
- *Monotonically-increase convergence case: $\forall i > 0, D_{i+1} \geq D_i$ and $\lim_{i\to\infty} D_i = HP_I$ when $\frac{2048}{HR} < D_0 \leq HP_I$*
- *Static convergence case: $\forall i > 0, D_i = D_0$ when $HP_I < D_0 < 2HP_I$*

Notice that since $\frac{2048}{HR} \ll 1$, we don't need to bother with the case of $D_0 \leq \frac{2048}{HR}$. Figure 3 depicts the convergence of $\{D_i\}$ according to Theorem 3.7. Note that Theorem 3.7 still holds even if HR changes afterward. In that case, the difficulty converges to $c \times HR(T)$, where $10 \leq c \leq 20$ and $HR(T)$ is the hash rate function over time. The proof for Theorem 3.7 is based on the same idea of Theorem 3.6, and the complete version is described in a supplementary technical report [26].

3.3 Irrelevance of Difficulty to KEPIs

We define the steady state of the blockchain as its PoW difficulty converges based on Theorem 3.7. The value of difficulty is independent of those blockchain KEPIs in a steady state.

In terms of Ethereum's block interval, $T_\infty \equiv D_\infty/HR$ become a constant between 10 to 20 seconds at convergence since D_∞

Location	Assumption
Equation 3	Mined blocks are always valid.
Equation 3	$V(D_i) \ll M(D_i)$ and thus $V(D_i)$ is ignored.
Hash Rate	$HR = 2^{26} = 64Mhps$ (hash per second) in Figure 2 and Figure 3, but is considered as a parameter in analysis.
Dilation Factor	$TDF = 0.1$ in virtual time, otherwise $TDF = 1.0$
Dilated Hash Rate	$HR^{VT} = HR \times TDF = 6.4Mhps$ in Figure 5, Figure 6, Figure 7, and Figure 8
Equation 1	Block number $i < 2 \times 10^5$ and thus $\lfloor 2^{\lfloor i/100,000 \rfloor - 2} \rfloor$ is ignored.
Equation 2	The computation power of the blockchain system is HR.

Table 1: Assumptions Made in the Analysis of the Difficulty in an Ethereum Blockchain.

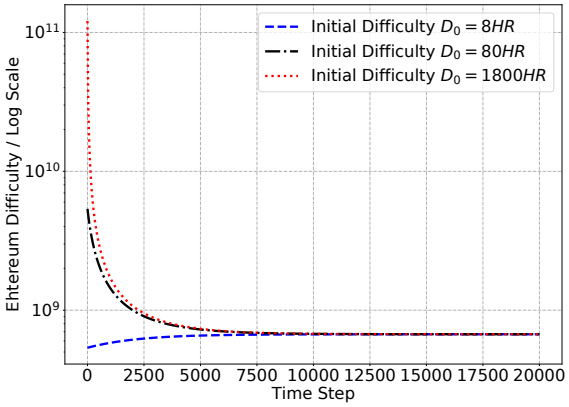


Figure 2: Based on Equation 7, Ethereum's difficulty converges to the implicit hash power HP_I regardless of the initial values.

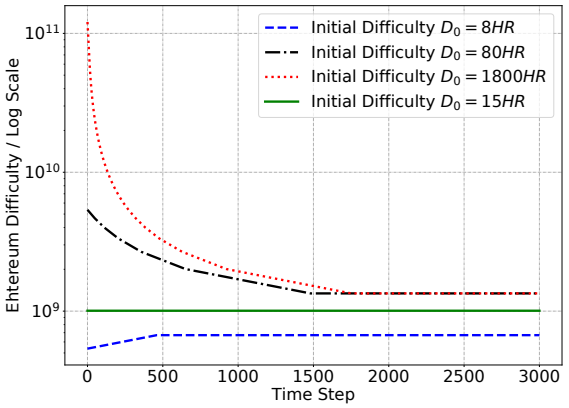


Figure 3: Based on Equation 8, Ethereum's difficulty converges in three ways depending on the initial value.

becomes $c \times HR$, $10 \le c \le 20$ according to Theorem 3.7; for Bitcoin, the mean block interval is fixed at 10 minutes, independent of the actual value of the difficulty.

In terms of transaction throughput π, between block b_i and block b_j $(i < j)$ in a converged blockchain

$$\begin{aligned} \pi(i,j) &= \frac{\#\text{ transactions stored in blocks}}{\sum_{k=i}^{j} T_k} \\ &\xrightarrow{\forall k \in [i,j], T_k = T_\infty} = \frac{\sum_{k=i}^{j} TX_k}{(j-i+1)T_\infty} = \frac{\overline{TX(i,j)}}{T_\infty} \\ &= \frac{\overline{TX(i,j)}}{c}, 10 \le c \le 20 \end{aligned} \tag{9}$$

where $\overline{TX(i,j)}$ is the average number of transactions stored in blocks. Again π is independent of the difficulty, but is determined by the number of transactions being put into blocks by ledger users. In terms of network latency and throughput, they are determined by the conditions of the underlying communication network, such as protocol stacks (e.g., TCP/UDP), hardware capabilities (e.g., bandwidth), network congestion (e.g., queuing) and other characteristics that are independent of the blockchain application layer. The analysis indicates that ignoring a converged PoW difficulty does not affect the fidelity of KEPIs. The fact actually speeds up the experiments when integrated with virtual time. Details are described in Section 5.

4 DESIGN OF MINICHAIN

The design goal of Minichain is to allow users to conduct high-fidelity emulation experiments of a blockchain system concerning both application behaviors and network characteristics at low cost (e.g., evaluating a new application design on a single laptop). There are two design objectives.

- Minichain enables realistic emulation of the blockchain applications with high usability, including direct execution of unmodified blockchain applications, flexible experiment generation, accurate emulation of transactions and mining activities and data collection.
- Minichain enables realistic emulation of the underlying network including user-specified delay, loss, and bandwidth per communication link.

Figure 4 depicts the two-layer system architecture corresponding to the two design objectives, and details are illustrated in the next two subsections.

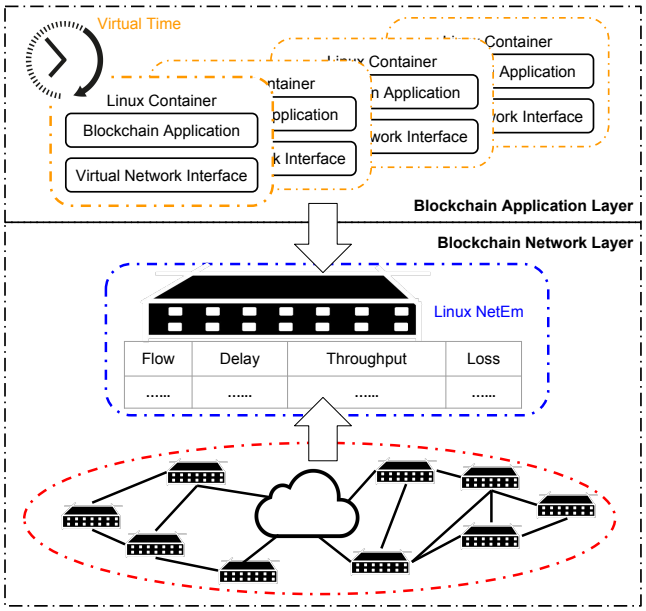


Figure 4: System Architecture of Minichain.

4.1 Emulation of Blockchain's Transactions and Mining Activities

Minichain supports the concurrent running of multiple blockchain peers to emulate a blockchain system. We conceptualize peers to be either blockchain application programs that commit transactions or PoW-based mining programs that help to confirm transactions with economic motivations (e.g., earning coins). Emulation of these peers requires Minichain to be

- Realistic: direction execution of the diverse and original blockchain codes conducting both transactions and mining activities;
- Transparent: execution of unmodified binaries;
- Extensible: support of various blockchain systems (e.g., Bitcoin, Ethereum and many other).

Minichain meets those requirements by running each peer indistinguishably within a Linux container [16]. The Linux container is a lightweight OS-level virtualization technique. Without additional efforts, users can directly execute unmodified binaries of blockchain peer regardless of the peer function (e.g., mining or transaction) or the type of blockchain (e.g., Bitcoin or Ethereum). A container offers a blockchain peer its own process tree, networking stack, and file system. Running peers inside containers is also resource-economic to enhance system scalability, as containers share the same kernel of the host machine.

While Linux container makes Minichain capable to run realistic, transparent, and extensible blockchain system, we encounter a unique challenge due to the characteristic of PoW in a blockchain. All miners work to solve mathematically hard puzzles (e.g., hashcash in Bitcoin and Ethereum). As a result, a significant amount of computational resources are consumed and it significantly increases the time for a blockchain system to converge, which negatively affects the performance of Minichain. However, the requirement of high-fidelity emulation forbids us to save emulation resources by eliminating or simplifying the mining activities. Therefore, the challenge Minichain confronts is how to reduce the time and resources spent on mining without modifying the mining code.

Our theoretical analysis of blockchain's convergence in Section 3 sheds lights upon this dilemma. Since the difficulty of PoW as well as the solving process of PoW do not affect the KEPIs after a blockchain converges, we propose to reduce PoW's difficulty by dilating the elapsed time from all miner's perspective. If solving a PoW takes a miner t seconds, we dilate the duration to $\frac{t}{TDF}$, where $TDF < 1$ is called time dilation factor. In this way, we have put peers in the virtual time of dilated speeds, a technique used by several network emulators [12, 18, 27], but for different purposes (typically $TDF > 1$). A blockchain running in virtual time still converges in the same way it does without virtual time (see the conclusion in Section 3), and only the converged difficulty is dilated by $\frac{1}{TDF}$. Moreover, the KEPIs (e.g., block interval and transaction throughput) are not affected by the dilated time. The proof and explanation to support the above statement are further elaborated in Section 5.

4.2 Emulation of Blockchain's Network Environment

Another design objective is to offer a realistic network environment to emulate blockchain applications. However, the scale of the blockchain's network is often too large to emulate in a commodity computer. In particular, it requires more resources than the underlying physical machine can offer. For example, Ethpool, a regular and healthy-functioning Ethereum mining pool, has more than 700 active workers [2]. Even emulating only the active miners in this pool requires bringing up hundreds of interconnecting network devices (e.g., switches and routers).

Considering the limited resources of a commodity machine (CPU, memory, bandwidth), we have to abstract the network emulation environment in Minichain in a way that well balances the emulation scalability and fidelity. Our guideline is to preserve the fidelity of the KEPIs described in Section 2.2 including the end-to-end network flow metrics such as delay, throughput, and packet loss between any two active blockchain peers. Meanwhile, other network performance metrics, like queuing delay on each device, interference among conflicting flows, are not the focus of our design. Under this guideline, we model a virtual link for any pair of blockchain peers that requires communication, where users can directly set the expected network flow characteristics. Implementation-wise, we utilize the Linux network emulation tool `NetEm` [17] to achieve the per-virtual-link configuration. The emulated network consisting of multiple virtual links can be abstracted as "one big switch" with active peers connected via one of its many ports. The traffic flow between any two peers is accurately shaped by `NetEm` according to the users' specification.

4.3 System Architecture

Minichain is composed of two layers as shown in Figure 4. In the blockchain application layer, blockchain peers are directly executed in the isolated Linux containers (orange boxes). Linux containers

support many types of blockchain systems, such as Ethereum and Bitcoin. The peers either run the mining code or make transactions as if they are on a real distributed blockchain system. By embedding the virtual time into all the containers, we are able to save both the computational resources consumed for blockchain's consensus algorithm as well as the emulation experimental running time.

The network layer produces the communication behaviors for the container-based blockchain peers. With the limited physical resources of the host machine, it is often infeasible to accurately emulate the large-scale underlying communication network (red oval). Minichain is designed to overcome this challenge by abstracting the physical network into a big switch entity (blue box). The containers are now logically connected in two steps. First, each container dedicates its virtual network interface to one port on the abstracted big switch. Second, users configure the network conditions of those ports using `NetEm`. In particular, for any active flow between two peers, we insert a flow entry describing the expected end-to-end delay, bandwidth, and packet loss. The design greatly enhances the system scalability without losing fidelity concerning those network layer KEPIs.

5 BLOCKCHAIN CONVERGENCE IN VIRTUAL TIME

The benefits to enable virtual time in Minichain include reduced computational resource and faster blockchain convergence speed when experimenting PoW-based blockchain applications. This section mathematically illustrates the benefits.

5.1 Convergence of Difficulty in Virtual Time

The block interval in virtual time can be expressed as

$$T_i = \frac{D_i}{HR \times TDF} = \frac{D_i}{HR^{VT}}$$

The *dilated hash rate* is defined with respect to the blockchain system's real hash rate, i.e., $HR^{VT} = HR \times TDF$. Under the same assumption that the contribution made by the block number is safely ignored, below is the modified version representing the difficulty dynamics.

$$D_{i+1} = D_i + \left\lfloor \frac{D_i}{2048} \right\rfloor \times \max(1 - \left\lfloor \frac{D_i/HR^{VT}}{10} \right\rfloor, -99) \quad (10)$$

The dividing line to eliminate the max function is the **dilated** explicit hash power, i.e., $D_i = HP_E^{VT} = 1000HR^{VT}$, and thus we obtain

$$D_{i+1} = \begin{cases} \beta^{VT} D_i^2 + \alpha_1 D_i, & D_i < HP_E^{VT} \\ \alpha_2 D_i, & D_i \geq HP_E^{VT} \end{cases} \quad (11)$$

We again replace the integer division with the normal division. Note that α_1, α_2 in Equation 11 are the same as those in Equation 4, but $\beta - \frac{1}{20480HR}$ in Equation 4 is adjusted to $\beta^{VT} = -\frac{1}{20480HR^{VT}}$. By repeating the same function analysis in Theorem 3.6, we draw the same conclusion of convergence as follows.

Theorem 5.1 (Convergence of Simplified Ethereum's Difficulty in Virtual Time). *Let us define the adjustment algorithm in **virtual time** as follows, where* $HR^{VT} = HR \times TDF$.

$$D_{i+1} = D_i + \frac{D_i}{2048} \times \max(1 - \frac{D_i}{10HR^{VT}}, -99) \quad (12)$$

We also define the dilated implicit hash power $HP_I^{VT} = 10HR^{VT} = 10HR \times TDF$. *Sequence* $\{D_i\}$ *converges to* HP_I^{VT} *regardless of the initial value. In other words,* $D_\infty \equiv \lim_{i\to\infty} D_i = 10 \times HR^{VT}$.

Theorem 5.1 is illustrated in Figure 5 given the constant hash power and $TDF = 0.1$. Note that the convergence value in virtual time ($D_\infty^{VT} = 64$ M) is only one tenth of the convergence value without virtual time ($D_\infty = 640$ M) shown in Figure 2.

Theorem 5.2 (Convergence of Ethereum's Difficulty in Virtual Time). *Let us define the adjustment algorithm in **virtual time** as follows, where* $HR^{VT} = HR \times TDF$.

$$D_{i+1} = D_i + \left\lfloor \frac{D_i}{2048} \right\rfloor \times \max(1 - \left\lfloor \frac{D_i/(HR^{VT})}{10} \right\rfloor, -99) \quad (13)$$

We also define the implicit hash power $HP_I^{VT} = 10HR^{VT} = 10HR \times TDF$. *Sequence* $\{D_i\}$ *converges in three different ways depending on the initial difficulty value. Specifically,* $\forall i \in \mathbf{Z}^+$:

- *Monotonically-decrease convergence.* $\forall i > 0, D_{i+1} \leq D_i$ *and* $\lim_{i\to\infty} D_i = 2HP_I^{VT}$, *when* $2HP_I^{VT} \leq D_0$
- *Monotonically-increase convergence.* $\forall i > 0, D_{i+1} \geq D_i$ *and* $\lim_{i\to\infty} D_i = HP_I^{VT}$, *when* $\frac{2048}{HR^{VT}} < D_0 \leq HP_I^{VT}$
- *Static convergence.* $\forall i > 0, D_i = D_0$, *when* $HP_I^{VT} < D_0 < 2HP_I^{VT}$

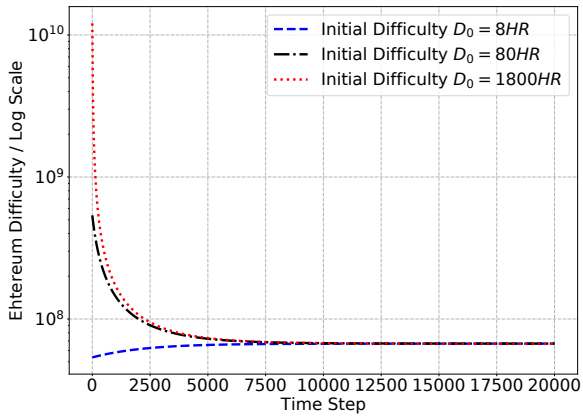


Figure 5: Based on Equation 12, Ethereum's difficulty converges in virtual time to a dilated implicit hash power regardless of the initial value.

Theorem 5.2 is illustrated in Figure 6 given the constant hash power and $TDF = 0.1$. Comparing to Figure 3, the convergence interval is reduced by 10 times from [640 M, 1,280 M] to [64 M, 128 M]. In summary, we have shown that virtual time, in theory, can linearly downscale the hash power needed for the Ethereum blockchain emulation to achieve a stable state.

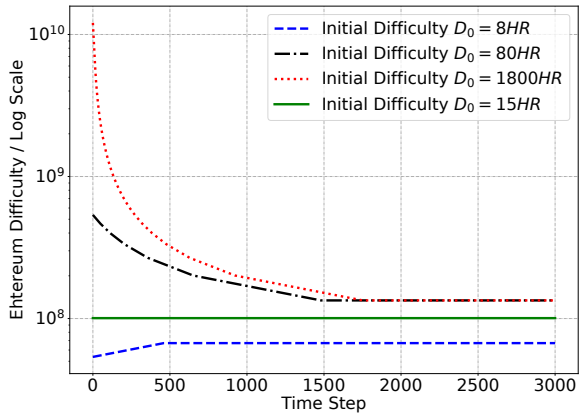


Figure 6: Based on Equation 13, Ethereum's difficulty converges in virtual time in three ways depending on the initial value.

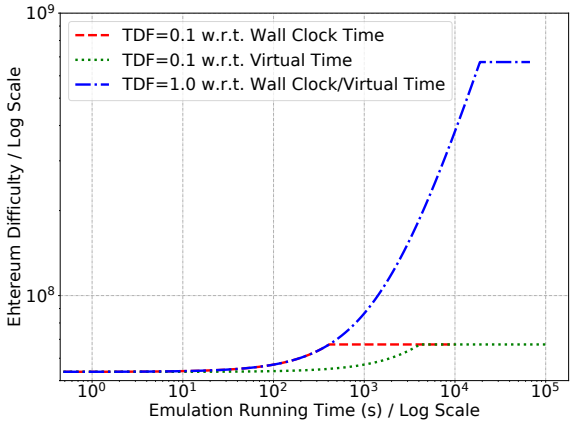


Figure 7: Based on Equation 8 and Equation 13 respectively, Ethereum's difficulty converges faster in virtual time with the fixed initial difficulty $D_0 = 8HR^{VT}$.

5.2 Fast Running Time for Convergence

We have so far analyzed the change of difficulty over time, it is also important to analyze the time for a blockchain emulation system to converge both with and without virtual time. We simulate two representative scenarios using the recursive relationship defined by both Equation 8 and Equation 13.

In the first scenario, we set the common initial difficulty to be $D_0 = 8HR^{VT}$ so that in both real time and virtual time cases, the difficulty will increase monotonically. We run the simulation for 10,000 steps to generate 10,000 blocks, the difficulty sequence, and the block generation timestamp sequence. Figure 7 plots the simulation results with both real time and virtual time. The second scenario is the same as the first scenario except that the common initial difficulty is set to $D_0 = 400HR^{VT}$. The difficulty monotonically decreases in both real time and virtual time as shown in Figure 8. Both scenarios together demonstrate that PoW in virtual time converges faster and to a lower difficulty value comparing to that in real time.

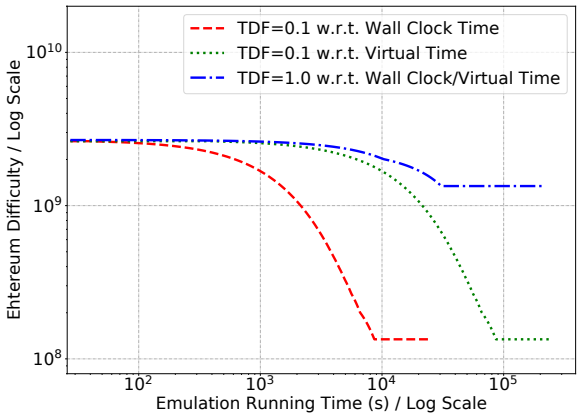


Figure 8: Based on Equation 8 and Equation 13 respectively, Ethereum's difficulty converges faster in virtual time with the fixed initial difficulty $D_0 = 400HR^{VT}$.

6 EXPERIMENTAL EVALUATION

We conduct the performance evaluation of Minichain concerning the blockchain application layer and the network layer respectively. We also evaluate the performance of virtual time. All experiments are executed on a Linux box with 20 cores (2.30GHz Intel Xeon CPU) and 32GB RAM. We use *Ping* and *iPerf3* to measure end-to-end network delay and throughput. We bring up the go-ethereum [7] blockchain application on each peer node emulated by Linux containers. From the perspective of blockchain applications, the underlying network forms a full mesh topology, i.e., a n-node network containing $\frac{n\times(n-1)}{2}$ communication links.

6.1 Blockchain Network Emulation

6.1.1 End-to-End Delay Error Analysis. Let us define $\Delta\delta$ to be the error of end-to-end delay between a pair of connected hosts. In particular, $\Delta\delta$ equals to the difference between the pre-configured link delay and the measured *ping* delay. We use *ping* to measure the delay among peers while varying the size of a fully connected mesh network using 2, 4, 8, 16 and 32 hosts. Each host pings every other host simultaneously every second, which results in $\frac{n\times(n-1)}{2}$ *ping* data flows. The error analysis of end-to-end delay is performed in three different scenarios, and we plot the experimental results using violin plots to show the full distribution of $\Delta\delta$ in Figure 9.

First, we measure the inevitable latency of Minichain by setting the delay of all links to zero and disabling mining activities in the blockchain. In Figure 9a, we observe that the inevitable latency on every link is ranging from 0.02 ms to 0.18 ms. The figure also shows that for all the links $\Delta\delta$ is close to the average value with small variances. For example, in the case of $n = 32$, the maximum and minimum delay errors are $\Delta\delta_{max} = 0.161$ and $\Delta\delta_{min} = 0.093$ respectively, both are close to the mean value 0.12. The results in

Figure 9a reveals a lower bound of the link delay Minichain that can accurately emulate on the testing machine.

Second, we re-use the same network and block application as the first scenario, but randomly set each link's delay to be uniformly distributed between 20 ms and 350 ms. We observe in Figure 9b that expected $\Delta\delta$s of each cases are around 0.1 ms with variances less than 0.02 ms. The error is at least two orders of magnitude less than the pre-configured link delay, thus has little impact on the performance of the blockchain applications above. In addition, the error does not increase as the size of the network increases, a desired feature for a scalable blockchain emulation testbed.

Third, we study whether the high-workload PoW mining tasks may affect the fidelity of network emulation in Minichain. Both miners and network emulation consume CPU resources, one for hash calculation and the other for data transmission. Figure 9c plots $\Delta\delta$ when we set half of the nodes to be heavy-duty miners. We observe the average $\Delta\delta$ for networks of different sizes are around 0.1 ms, i.e., approximately 0.8% of the expected link delay (20 ms), which again demonstrates the small error and good scalability of Minichain regarding end-to-end network latency. In addition, the high CPU usage for mining does not increase the latency error as the $\Delta\delta$ is in the same range compared with Figure 9a and Figure 9b.

6.1.2 Ene-to-End Throughput Error Analysis. We conduct an error analysis of end-to-end throughput by measuring the throughput among blockchain peers and comparing with the user-defined bandwidth in various scenarios.

We select n hosts in a blockchain and bring up a UDP server and a UDP client on each host. Each host sends a UDP flow to every other host using *iPerf3* and together there are $n \times (n-1)$ concurrent UDP communication flows. One constraint is that the total throughput cannot exceed the hardware limit of the physical machine (e.g., 50 Gbps in our system). Therefore, we adjust the network size for different link bandwidths to meet this requirement. For example, given a full-mesh topology, if the bandwidth between two hosts is set to 1 Gbps, the total number of links must be smaller than or equal to 50 (i.e., 50 Gbps/1 Gbps), and the total number of hosts is at most 7. We denote $\Delta\tau$ as the difference between the measured throughput τ and the pre-configured link bandwidth B.

First, we set the bandwidth of all links to 10 Mbps and disable all mining activities in the blockchain. Figure 10a shows the distribution of $\Delta\tau$ with different network sizes. As the number of hosts increases exponentially from 2 to 32, we observe that every link's $\Delta\tau$ is around 0.05 Mbps, i.e., only 0.5% of the line rate. The variance of $\Delta\tau$ increases but the range is still acceptably small, e.g., even when $n = 32$, 95% of the errors fall within 0.035 ± 0.005 Mbps. The results indicate that Minichain produces accurate network throughput for duplex emulated links, and the performance of throughput remains accurate as the network size grows.

Second, we increase the link bandwidth to 100 Mbps and repeat the same tests. Note that due to the hardware limit, we can emulate no more than 500 100 Mbps links (without virtual time). For a full mesh topology, the number of nodes cannot exceed 22, i.e., $n \times (n-1) < 500$. As a result, we show the results of different network sizes up to 16 hosts. In Figure 10b, we observe that $\Delta\tau$ is distributed around 0.4 Mbps except for the $n = 16$ case. Minichain achieves 99% of the required throughput when emulating 100 Mbps links. When the size of the full mesh network increases to 16 (close to the resource limit), the average throughput error deteriorates to 6.64 Mbps, around 6.6% deviates from the pre-defined bandwidth, which is still reasonably good for many blockchain testing applications.

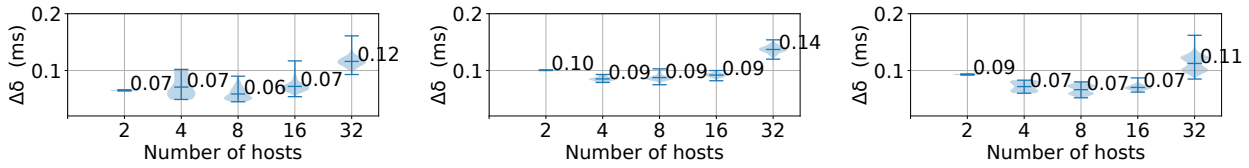
Third, we conduct the throughput error analysis with high-workload PoW mining tasks by setting half of the hosts to miners, and plot the results in Figure 10c. The link bandwidth is set back to 10 Mbps, a reasonable value in real-world applications. Note that in a blockchain network, nearly all of the bandwidth resources are used for broadcasting and downloading blocks. For example, the maximum size of a block in Ethereum is 780 KB, and a 10 Mbps link is sufficient to support the average block interval of 10 to 19 seconds. In Figure 10c, we observe that $\Delta\tau$ is in the range of [0.01, 0.025] Mbps. Even the number of hosts is 32, $\Delta\tau$ is strictly bounded by 0.03 Mbps. The results indicate that the fidelity of the network is not affected when Minichain spends a significant amount of computational resources on blockchain applications.

In summary, Minichain is well designed to support emulation of a blockchain network with user-specified per-link delay and bandwidth. The actual delay and throughput are very close to the desired ones as long as we do not overwhelm the resources of the underlying physical machine. Even on one commodity server, we can emulate a blockchain network with decent size. Note that two untrusted parties in a real-world blockchain system generally have no reason to establish a high-bandwidth connection. Furthermore, as Linux containers isolate the inside blockchain applications from the kernel-level `NetEm`, we observe little interference between the running applications and the underlying emulated network.

6.2 Block Application and Virtual time

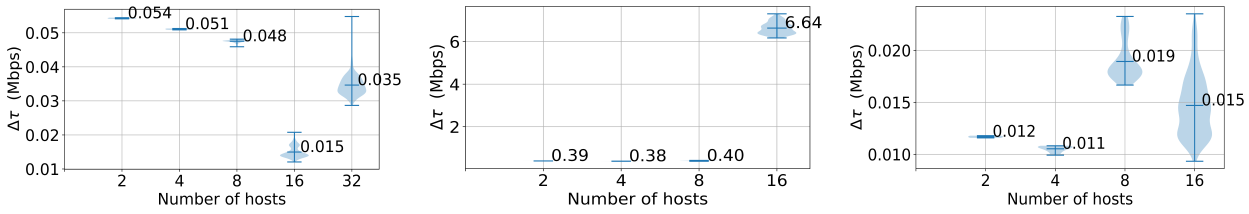
We evaluate the performance of blockchain application emulated by Minichain as well as the impact of virtual time. We choose to measure two block-related KEPIs, average block interval, and block distribution. The block interval is closely related to the throughput of transaction operation as shown in Equation 9. The ideal block interval is mathematically defined in the consensus algorithm. Performance of an emulated blockchain can thus be evaluated by measuring the block interval and then comparing it with the theoretical value. The block distribution represents how the rewards (e.g., Bitcoins) are spread out in the blockchain system and received by the miners. Ideally, the number of rewards a miner obtains should be proportional to the number of computational resources the miner invests in the long run. Therefore, an accurate emulated blockchain system should distribute the blocks to miners proportional to the CPU resource allocation.

We emulate an Ethereum blockchain system containing 6 miners in Minichain. Miners start PoW-based mining with the minimum difficulty, 131,072. We assume that one miner owns 30% of the CPU resources and the other five miners evenly share the remaining CPU resources (i.e., 14% for each miner). The six miners form a full-mesh network topology. We set the communication link between each pair of miners to be 10 Mbps with 20 ms delay. We run the same sets of experiments without virtual time and with virtual time for exactly 80 hours of virtual time and make sure the Ethereum system converges in both cases. We present the resultant performance metrics in Table 2 for comparison. "Execution Time in Wall Clock"



(a) δ = **0 ms, no miners.** **(b)** $\delta \sim$ **U[20ms, 350ms], no miners.** **(c)** δ = **20 ms, 50% mining hosts.**

Figure 9: Error Analysis of End-to-End Delay. $\Delta\delta$ **is the difference between the pre-configured link delay** δ **and the measured *ping* delay.**



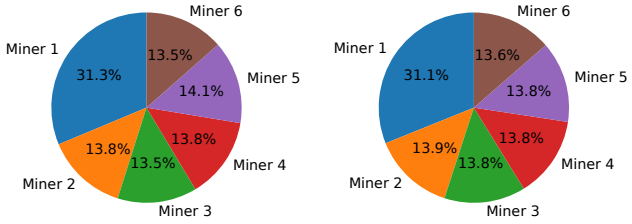
(a) B = 10 Mbps, no miners. **(b) B = 100 Mbps, no miners.** **(c) B = 10 Mbps, 50% mining hosts**

Figure 10: Error Analysis of End-to-End Throughput. $\Delta\tau$ **is the difference between the measured throughput** τ **and the pre-configured link bandwidth B.**

represents the actual time taken to finish the experiment. "Height of Chain" is the number of blocks generated during the emulation. "Mean Block Interval" is calculated as the height of chain divided by virtual running time (i.e., 80 hours). We also plot the block distribution for both cases in Figure 11.

Figure 11a shows the distribution of blocks without virtual time. The expected behavior is that the block allocation among the miners should be proportional to their allocated CPU resources, and we observe such allocation in Figure 11a. In other words, like in a real blockchain system, the probability of publishing the next block matches a miner's computing capacity in Minichain. In the entire experiment, 22,307 blocks are generated with an average block interval of 12.9 seconds.

Figure 11b shows the distribution of blocks with virtual clock and $TDF = 0.1$. When the experiment finishes, the results with virtual time generate nearly the same distribution shown early in Figure 11a. While producing the same emulation results (e.g., the average block interval oscillates between 10 and 19 seconds), we observe that Minichain is able to complete the emulation 10 times faster than the case without using virtual time, as shown in the third column in Table 2. In addition, virtual time adjusts the converged difficulty in Ethereum to a smaller value. The impact is that the number of blocks generated in virtual-time-enabled emulation is 1,005 greater than the one generated without virtual time. Even so, the mean block intervals for both cases are still very close, differing by less than 1 second. The reason is that the 80-hour emulation runs long enough so that the duration of convergence is dominating the process in which the difficulty keeps increasing. In summary, virtual time helps Minichain to significantly reduce the experimental execution time with little fidelity loss.



(a) No Virtual Time **(b) Virtual Time, TDF = 0.1**

Figure 11: Distribution of Blocks among Six Miners in Minichain with and without Virtual Time.

Metrics	No Virtual Time	$TDF = 0.1$
Execution Running Time	80 h	8 h + 2.7 s
Height of Chain	22,307	23,312
Mean Block Interval	12.9 s	12.3 s

Table 2: Blockchain Performance Metrics with and without Virtual Time.

7 CASE STUDY: NETWORK IMPACT ON SELFISH MINING

We present a case study of Minichain by realizing the selfish mining attack scenario described in [29]. The selfish mining attack [9] leverages the rule that a blockchain system treats the longest chain as its canonical one. It is sometimes also known as 51% attack because it is usually launched by a miner, or a pool of miners, that controls more than 50% of the total computational power inside a blockchain network. With its dominant resource over the blockchain network,

the attacker eventually generates a chain of valid blocks longer than others, monopolizing the blockchain system that ought to be decentralized. Once realized, the selfish mining attacker is able to determine which new transaction gains confirmation or is able to reverse confirmed transactions to double-spend coins.

However, if one takes the network latency into account, a malicious miner with less than 50% hashrate still has the potential to damage a healthy blockchain in the way similar to selfish mining [29]. To demonstrate the feasibility of such an attack and evaluate its impact, we create a blockchain consisting of 6 fully-connected miners in Minichain. Miner X owns 45% CPU resource of the test machine, and the other five peers share the remaining 55% evenly. We emulate the blockchain with an average block interval of one second (technically achieved by setting a favorably low difficulty) and launch the attack. We stop the emulation when the 1000-th block is produced. We repeat the emulation experiment six times and the results are plotted in Figure 12.

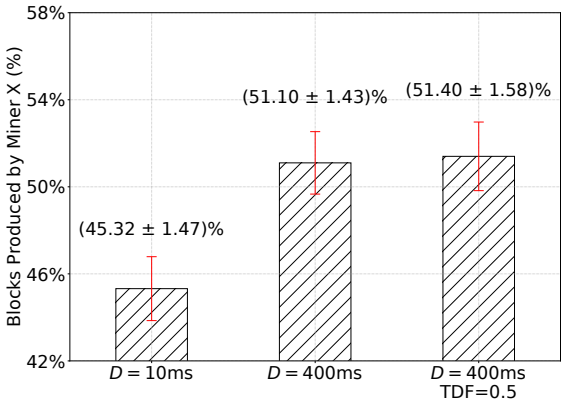


Figure 12: In a 6-miner blockchain system, miner X possesses 45% CPU resources. With different network delay (δ) settings, miner X's final distribution of blocks varies. In a high-latency network (δ = 400 ms), miner X successfully dominates the blockchain (owning >50% of blocks). Repeating the experiment with virtual time (TDF = 0.5) produces the same behavior but shortens the execution time.

The left bar in Figure 12 shows miner X's share of blocks when we set all link delays to 10 ms. We observe that miner X did not produce more than half of the blocks in the system for all runs, i.e., only 45.32% ± 1.47% (mean and standard deviation) of the blocks. However, after setting the link delay to 400 ms, miner X generates more than half of the blocks in five out of six runs. The blocks generated by miner X is 51.10% ± 1.43%, as shown in the middle bar in Figure 12. With the high network latency and low block interval, it is highly likely (83.33%) that miner X monopolizes the blockchain with less than 50% of the entire computation power. Minichain is able to demonstrate the 51% attack scenario by emulating a realistic networking environment. Considering the effect of network conditions often reveals unexpected behavior of a blockchain system, especially when cyber-security is a critical concern. Minichain is thus designed to fill this gap for blockchain testing and evaluation.

With the average block interval being 1 second and the number of generated blocks being 1000, each of the six repeated low-latency-emulation experiments takes 1014 ± 6.19 seconds to complete; for the six high-latency-network emulations, it takes 1015 seconds on average for every one of them. We now enable the virtual time feature in Minichain with the goal of accelerating the case study experiment without losing fidelity. The right bar in Figure 12 shows miner X's share of blocks over six runs by setting link delay to 400 ms and TDF to 0.5, i.e. time elapsed two times faster than the wall clock. We observe that the result (51.40% ± 1.58%) is almost the same as the case without virtual time (i.e., the middle bar). In five of six runs, miner X generates more than half of the blocks in virtual time. Meanwhile, with $TDF = 0.5$ Minichain shortens the experiment running time by nearly half (on average 507.5 seconds with ±4.13 standard deviation).

8 RELATED WORKS

8.1 Blockchain Testing and Evaluation

Simulation and public testnet are the two most common methods for testing and evaluating blockchains and their applications at present. Researchers have proposed multiple simulation frameworks to analyze blockchains including (1) security and performance of PoW [10], (2) scalability [11], (3) transaction latency [28], and (4) quality of network infrastructure [25]. Shadow [14], a parallel discrete-event network simulation platform, provides a plug-in [21] to support simulating thousands of Bitcoin reference clients over a large-scale network topology. While simulation offers great scalability and flexibility, people often question the fidelity since simulation executes an abstracted model of the target system to produce behavior. Minichain, in contrast, 1) allows direct execution of unmodified blockchain application programs in Linux containers and can be easily extended to support various blockchain systems, while Shadow requires source code modification of the`bitcoind` software; 2) runs real network stack inside the Linux kernel to emulate network activities; 3) integrates the virtual time to container-based emulation for performance enhancement, while [21] uses simulation time.

Testnet is an alternative blockchain network maintained by volunteers in the blockchain community to test their smart contracts on a test network before deployment on the main blockchain. It provides high fidelity as the testnet is a replica of the main chain except testnet uses another set of configuration parameters. However, testnet does not offer sufficient controllability and flexibility to users. First, users cannot customize the network conditions of testnet and thus unable to evaluate the impact of network environment on blockchain applications. It is also ethically wrong to introduce unwanted interference to a shared testnet by reproducing known attacks in it. Minichain, in contrast, is owned exclusively by each user. Users have the great flexibility to generate customized testing scenarios and have tight control over them during the experiment. Another line of research develops private evaluation frameworks for blockchain systems [5, 24]. Such frameworks benchmark the performance of a blockchain located in a private cluster with dozens of physical machines. They do not offer interfaces for users to configure the network environment. Moreover, running experiments on a private cluster involves manual hardware configuration.

8.2 Virtual Time for Network Emulation

Virtual time has been well studied to improve the performance of network emulation. The time-dilation-based network emulation was first proposed by [12] and implemented in Xen to scale CPU and network bandwidth for network protocol and application testing. It has also been studied in the software-defined network emulator Mininet [20] for fidelity and scalability enhancement and various simulators [13, 15, 19] to address the challenge of synchronization between emulation and simulation. However, the existing works focus on improving the temporal fidelity when the network emulated exceeds the system physical resources. TDF is typically set to be greater than 1, and thus slowing down Linux processes execution time to trade for the accurate emulation results. In this work, Minichain utilizes the virtual time technique in a different way. Specifically, we set TDF to be less than 1 in order to speed up the execution time of emulated blockchain peers with the effect of scaling down the difficulty in the PoW consensus algorithm.

9 CONCLUSION

We present Minichain, a Linux-container-based blockchain emulator for testing and evaluating blockchain applications on a commodity computer. The unique characteristics of Minichain include a realistic and configurable network emulation environment and a virtual time system to speed up the experiment execution without losing fidelity. In the future, we will utilize Minichain to performance cyber-security evaluation of blockchain systems and applications. We will also further enhance the scalability of Minichain to support distributed emulation, and compare its performance with cluster-based blockchain testbeds.

ACKNOWLEDGMENT

This work is partly sponsored by the Air Force Office of Scientific Research (AFOSR) under Grant YIP FA9550-17-1-0240 and the Maryland Procurement Office under Contract No. H98230-18-D-0007. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFOSR and the Maryland Procurement Office.

REFERENCES

[1] Ian Altman. 2018. How Blockchain Will Transform Business And The Law. (June 2018). Retrieved Janunary 10, 2019 from https://www.forbes.com/sites/ianaltman/2018/06/29/blockchain-changes-business-law/#611e39135cb9

[2] Bitfly. 2018. Ethpool - Info. (Jan. 2018). Retrieved Janunary 5, 2019 from https://ethpool.org

[3] Jacob Boersma. 2018. 5 blockchain technology use cases in financial services. (June 2018). Retrieved Janunary 10, 2019 from https://www2.deloitte.com/nl/nl/pages/financial-services/articles/blockchain-technology-use-cases-in-financial-services.html

[4] Danny Crichton. 2018. Liquid democracy uses blockchain to fix politics, and now you can vote for it. (Feb. 2018). Retrieved Janunary 10, 2019 from https://techcrunch.com/2018/02/24/liquid-democracy-uses-blockchain/

[5] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. Blockbench: A framework for analyzing private blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1085–1100.

[6] ECoinmerce. 2018. Tokenized E-commerce. (Jan. 2018). Retrieved Janunary 10, 2019 from https://www.ecoinmerce.io

[7] ethereum. 2017. go-ethereum. (Jan. 2017). Retrieved Janunary 5, 2019 from https://github.com/ethereum/go-ethereum/

[8] ethereum/EIPs. 2018. EIPS/eip-2.md. (Jan. 2018). Retrieved Janunary 5, 2019 from https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.md

[9] Ittay Eyal and Emin Gün Sirer. 2013. Majority is not Enough: Bitcoin Mining is Vulnerable. *CoRR* abs/1311.0243 (2013). arXiv:1311.0243 http://arxiv.org/abs/1311.0243

[10] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 3–16.

[11] Sneha Goswami. 2017. Scalability analysis of blockchains through blockchain simulation. (2017).

[12] Diwaker Gupta, Kenneth Yocum, Marvin McNett, Alex C. Snoeren, Amin Vahdat, and Geoffrey M. Voelker. 2005. To Infinity and Beyond: Time Warped Network Emulation. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP '05)*. ACM, New York, NY, USA, 1–2. https://doi.org/10.1145/1095810.1118605

[13] Christopher Hannon, Jiaqi Yan, and Dong Jin. 2016. DSSnet: A Smart Grid Modeling Platform Combining Electrical Power Distribution System Simulation and Software Defined Networking Emulation. In *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '16)*. ACM, New York, NY, USA, 131–142. https://doi.org/10.1145/2901378.2901383

[14] Rob Jansen and Nicholas Hooper. 2011. *Shadow: Running Tor in a box for accurate and efficient experimentation*. Technical Report. MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE AND ENGINEERING.

[15] Dong Jin, Yuhao Zheng, Huaiyu Zhu, David M Nicol, and Lenhard Winterrowd. 2012. Virtual time integration of emulation and parallel simulation. In *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, 201–210.

[16] Michael Kerrisk. 2013. Namespaces in operation, part 1: namespaces overview. (Jan. 2013). Retrieved Janunary 5, 2019 from https://lwn.net/Articles/531114/

[17] Michael Kerrisk. 2018. Man Page for NETEM - Network Emulator. (Jan. 2018). Retrieved Janunary 5, 2019 from http://man7.org/linux/man-pages/man8/tc-netem.8.html

[18] Jereme Lamps, Vignesh Babu, David M. Nicol, Vladimir Adam, and Rakesh Kumar. 2018. Temporal Integration of Emulation and Network Simulators on Linux Multiprocessors. *ACM Trans. Model. Comput. Simul.* 28, 1, Article 1 (Jan. 2018), 25 pages.

[19] Jereme Lamps, Vignesh Babu, David M Nicol, Vladimir Adam, and Rakesh Kumar. 2018. Temporal Integration of Emulation and Network Simulators on Linux Multiprocessors. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28, 1 (2018), 1.

[20] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A network in a laptop: rapid prototyping for software-defined networks. ACM Press, 1–6. https://doi.org/10.1145/1868447.1868466 01341.

[21] Andrew Miller and Rob Jansen. 2015. Shadow-Bitcoin: Scalable Simulation via Direct Execution of Multi-Threaded Applications. In *8th Workshop on Cyber Security Experimentation and Test (CSET 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/cset15/workshop-program/presentation/miller

[22] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[23] The Monero Project. 2018. MONERO: Private Digital Currency. (Feb. 2018). Retrieved Janunary 10, 2019 from https://www.getmonero.org

[24] Huawei Technologies. 2017. Caliper: A Blockchain Benchmark framework. (Jan. 2017). Retrieved Janunary 5, 2019 from https://github.com/hyperledger-archives/caliper

[25] Bozhi Wang, Shiping Chen, Lina Yao, Bin Liu, Xiwei Xu, and Liming Zhu. 2018. A Simulation Approach for Studying Behavior and Quality of Blockchain Networks. In *International Conference on Blockchain*. Springer, 18–31.

[26] Jiaqi Yan. 2018. On Convergence of Ethereum's Difficulty under Virtual Time. (Janunary 2018). Retrieved Janunary 3, 2019 from https://drive.google.com/file/d/15iK1eDGLwShPoXYH3_0gQ9enSoiDSK_U/view?usp=sharing

[27] Jiaqi Yan and Dong Jin. 2015. VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15)*. ACM, New York, NY, USA, Article 27, 7 pages. https://doi.org/10.1145/2774993.2775012

[28] Rajitha Yasaweerasinghelage, Mark Staples, and Ingo Weber. 2017. Predicting latency of blockchain-based systems using architectural modelling and simulation. In *Software Architecture (ICSA), 2017 IEEE International Conference on*. IEEE, 253–256.

[29] Qi Trey Zhong and Zak Cole. 2018. Analyzing the Effects of Network Latency on Blockchain Performance and Security Using the Whiteblock Testing Platform. (Jan. 2018). Retrieved Janunary 5, 2019 from https://www.whiteblock.io/library/analyzing-effects-network.pdf