A Uniform Model for Analyzing Efficiency of PBFT-based Blockchain

Zhaojin Li
The University of Queensland
Brisbane, Australia
zhaojin.li@uq.net.au

Naipeng Dong
The University of Queensland
Brisbane, Australia
n.dong@uq.edu.au

Raghavendra Ramesh
SupraOracles
Brisbane, Australia
r.ramesh@supraoracles.com

Guangdong Bai
The University of Queensland
Brisbane, Australia
g.bai@uq.edu.au

***Abstract*—The PBFT (Practical Byzantine Fault Tolerance) algorithm has been widely adopted in large amounts of blockchain consensus mechanisms, due to its better performance compared to traditional Proof-of-Work mechanisms. Therefore, many studies have been conducted on their performance analysis particularly the efficiency of reaching a consensus. However, existing analyses are largely ad-hoc and specific to the running environment of particular blockchains (e.g., hardware specifications and network configurations), such that using their analysis results for benchmarking and comparison among different blockchains is unfair. Hence, in this paper, we propose a measurement of blockchain efficiency independent of the running environment. To do so, we abstracted the process of the PBFT-based blockchain, and identified factors that impact the efficiency. Based on this, we built a probabilistic model to evaluate two efficiency properties of PBFT-based blockchain, i.e., the network load and the finality. Through the model, we investigated the relations of the factors and efficiency as well as the relations among the factors.**

***Index Terms*—Blockchain, PBFT, Efficiency, Finality, Consensus, Bayesian network**

I. Introduction

Blockchain is attracting an increasing investment which is expected to reach 176 billion dollars in 2025 and up to 3.1 trillion dollars by 2030 further [25]. The Practical Byzantine Fault Tolerance (PBFT)-based blockchain, as an important part [28], has witnessed an increasing number of blockchains, due to its high performance, scalability as well as versatility [21]. For example, PWC (PricewaterhouseCoopers) [5] lists a variety of usages on business and economy in the future with a potential market size of up to 1.76 trillion dollars.

Many existing PBFT-based blockchains have claimed high performance measured by efficiency properties like *throughput* which measures the transactions that can be handled per second and *finality* which measures the time cost for a transaction to be confirmed. For example, Solana claims that its maximum throughput can reach $700,000/s$ [29]. However, such claims are sometimes questionable. First, the actual throughput observed is way smaller than the claimed. For example, the real-time throughput of Solana was only around $2,000/s$ (observed at its official dashboard [6]), i.e., the max throughput may be never reached. Second, the finality observed varies depending on the running environment (see Solana Dashboard [6]), meaning that the finality may deviate drastically in extreme cases. However, there is no evaluation on such cases. After investigation, we found that the claimed high throughput and finality depend on its hardware and network configurations. For instance, Hedera only performs at a high level of efficiency when there are a large number of users (see Section VII).

There have also been academic studies analyzing the performance of (PBFT-based) blockchains, focusing on throughput and latency—a similar but slightly different definition of finality. Notably, previous studies [15], [18], [23], [24], [30], [31] analyze the throughput and latency by simulating the target blockchain. However, they suffer from the following limitations: 1) Existing works are mostly ad-hoc and blockchain-specific, hindering them from being adopted to analyze others. 2) They all are based on simulation which is limited by the small number of nodes. 3) Blockchains cannot be fairly using them, since their analysis results are obtained in different environments with different hardware and network configurations.

Therefore, to facilitate a trustworthy evaluation and fair comparison across blockchains, a uniform framework independent of the running environment of PBFT-based blockchain is needed to evaluate the performance. To this end, we identified a set of key factors to represent the running environment. Taking these key factors as hyper-parameters, we build a model to theoretically calculate the performance of a blockchain. In particular, we focus on analyzing two efficiency properties, i.e., *message complexity*—the number of messages needed to reach consensus, named *network load*, and *time complexity*—the number of message hops to confirm a transaction, named *finality*. Note that this work is analysis-based instead of experiment-based, i.e., no real-time measurement (e.g., per second, in second) can be used. Therefore, we exclude measuring throughput and latency. Instead, we use the number of messages and message hops to measure efficiency.

To build a generic analysis model, we abstracted the process of PBFT, and formed a probabilistic process based on the Bayesian network [26] to calculate the message cost when reaching any particular state before the consensus is reached. On top of this, we formulate the two efficiency properties as probabilistic conditions for reaching consensus and finality. Using this model, we quantitatively analyzed the impacts of key factors on efficiency, and the relations among the factors. Our analysis yielded 10 key findings to facilitate blockchain designers, developers and users when they design, customize or apply blockchains.

Contributions. The main contributions of this work can be summarized as follows:

1) To the best of our knowledge, we are the first to provide a uniform evaluation model of PBFT-based blockchain that facilitates fair cross-blockchain benchmarking and comparison.
2) This work provides new insights and knowledge of blockchain efficiency to support decision making on blockchain deployment and applications. This is achieved by defining a set of key factors and showing the relationship between the key factors and efficiency.
3) We performed case studies to confirm the accuracy of our model by applying our model to real-world blockchains, and demonstrated the usage of our model in uncovering unrevealed information about the blockchains.

II. Background

In this section, we introduce the essential concepts used in the subsequent sections.

A. Practical Byzantine Fault Tolerance

The general process of the PBFT-based consensus [13] can be illustrated by the following stages of confirming a single block as shown in Fig. 1 where different colors of arrows represent messages in different stages.

1) Pre-prepare stage: The leader node collects transactions and forms a new block, signs this block, and then sends it to other nodes following the corresponding gossip protocol. All other nodes wait for the block from the leader node.
2) Prepare stage: On receiving a signed block (named as an approval) from other nodes, a node immediately verifies whether the block is redundant and whether the signature and the block are valid. If so, the node signs the block, transfers its approval to others, and increases the count of this block by 1. Once a node has received supermajority ($\frac{2}{3}$ of total nodes or more) approvals from other nodes, it enters the next stage.
3) Commit stage: A node sends a commitment message of the block to other nodes. When a node receives a supermajority of commitments, meaning that the block has been committed by a supermajority of nodes, this node will add the block to its local blockchain. Then the node goes to the next iterations.

When the supermajority of nodes added this block to its local blockchain, it is called that the consensus/finality of this block is achieved. And the finality in PBFT is deterministic, contrary to Bitcoin where its finality is probabilistic (with only a negligibly small probability a block can be changed).

B. Gossip Protocol

The gossip protocol is used for all nodes in a network to send messages during a given *gossip frequency* (default by 10/s) in a peer-to-peer manner. Each non-redundant message

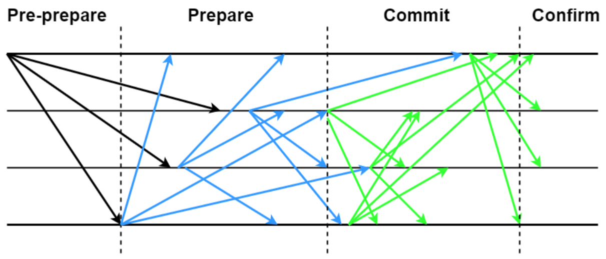


Fig. 1. A generic PBFT process

is analogous to a rumor, meaning that a node transfers the received messages to others to spread the rumor. A node randomly selects a set of nodes to spread the rummer. The number of selected nodes is called the *fanout*. A node in the protocol can be at one of the 3 states: Susceptible (S): The node does not know the rumor; Infected (I): The node knows the rumor and is actively spreading it; Removed (R): The node has seen the rumor, but is not participating in the spreading. This is called the SIR model [22]. Based on this model, there are the following two broadcasting mechanisms.

- SIR gossip (aka rumor-mongering). The node is initially in the S state. On receiving a new rumor, its state transforms to I, and the node starts to spread the rumor. Once finishing spreading the rumor, its state transforms to R. The advantage of SIR gossip is its low load on the CPU and network, because it only spreads the newest information, and nodes actively push messages to other nodes, which is called 'gossip-push'.
- SI gossip. Only the S and I states are involved in the SI gossip i.e., no R state. A node (at any state) requests to sync messages with all others, called the 'gossip-pull'. On request, an infected node spreads both the newest messages and historical messages. The advantage of SI gossip is that it can transmit a larger amount of information. The disadvantage is its cost of resources.

C. Network Models

In this paper, we assume a semi-synchronous network, namely there is a fixed upper bound Δ on the time required for a message to be sent from one node to another and a fixed upper bound Φ on the relative speed (one processor's clock can run faster than another's), following a popular definition in [16]. The Δ and Φ are not priori known but can be measured. Note that the original term used in [16] is *partial synchrony*. We adopt the recent terms *semi-synchronous*, which is widely used the in the literature [14].

D. Bayesian Network

The Bayesian network [26] is a type of probabilistic graphical model that uses Bayesian inference for probability computations. A Bayesian network is a directed acyclic graph consisting of states and labeled transitions (see an example in Fig. 2). The labeled transitions represent the dependencies between states, where the source state is the parent state

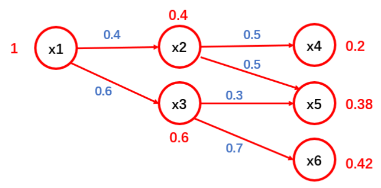


Fig. 2. A simple probability distribution graph

and the destination state is the child state. The label is the probability of the child being dependent on its parents. The states without an arrow between them are independent.

Bayesian networks satisfy the local Markov property—a state is conditionally independent of its non-descendants given its parents. This property allows us to simplify the calculation of each state, as it only depends on its parent states. This is illustrated using Fig. 2. In the figure, the red values are the probabilities of states, and the blue values on the arrows are the probabilities of the state transformation. The probability of a state can be calculated as the sum of its conditional probabilities i.e., the probabilities of its parent states, using the following formula:

$$Pr(X = x) = \sum Pr(X = x \mid Y = y_i) * Pr(Y = y_i) \quad (1)$$

where y_i is a conditional probability of x. For example, Pr_{x5} = Pr_{x2} * 0.5 + Pr_{x3} * 0.3 = 0.4 * 0.5 + 0.6 * 0.3 = 0.38.

III. Related Works

Most existing works on evaluating the efficiency of blockchains focus on a specific platform, for example Ethereum [11], Hyperledger Fabric [9], and Parity [27]. These works cannot be adopted to compare different blockchains. There are notably a few works that analyze multiple blockchains, which are summarized as follows:

Hao et al. [18] proposed an evaluation method of blockchain performance including throughput and latency, for two blockchain platforms: Ethereum and Hyperledger Fabric. It deploys blockchains locally and simulates 4 nodes and clients. Then a simple smart contract is deployed which transfers tokens in a given set of frequencies. The authors collect the number of successful transactions and their latency. Its experiments showed that the performance of PBFT (Hyperledger) outweighed PoW (Ethereum) in all experiment cases, and demonstrated that consensus plays a key role in their performance. Similar research was conducted by Suporn et al. [24] on the same blockchains against the same performance properties. Suporn et al. additionally proved that Ethereum is able to handle more concurrent transactions.

Santosh et al. [23] proposed an analysis tool to evaluate the system stability (defined as the needed time to achieve consensus for a block) and throughput of blockchains. The authors defined different evaluation scenarios aiming to find the suitable block size to allow the system to run stably, and tested the impact of its stability with the change of the number of nodes. Finally, this work compared the analyzed results with the real-world blockchains and demonstrated how to apply that tool via a real-world example.

Peilin et al. [31] proposed a framework to monitor the performance of 4 types of consensus including PoW, PoS, and PBFT. It proposed detailed performance metrics focusing on memory cost and CPU cost, and a log-based monitoring framework. The authors use a set of smart contracts as test cases to monitor the performance of the 4 blockchains.

Tien et al. [15] proposed a benchmark framework to analyze blockchains. Through simulation using the framework, a performance baseline of blockchains is provided for various numbers of nodes (up to 32), including throughput, latency, scalability, and fault-tolerance. This is used to evaluate the efficiency of Ethereum, Parity, and Hyperledger Fabric.

These works are summarized in Table I, including their target blockchains, evaluation metrics, and analysis methods. It is obvious that they all focus on specific blockchains and cannot be adopted for comparison between any blockchains. And they are all virtual-machine-simulation-based approaches with limited nodes, hence depending on their specific running environments and having limited scalability.

IV. PBFT-based Blockchain Abstraction

Addressing the above limitations requires a uniform model to evaluate the efficiency of PBFT-based blockchains. To this end, we first identified the key factors to represent the running environment, named the *Static Settings* (Section IV-A) and abstracted the PBFT processes, named the *Behavior Model* (Section IV-B). This serves as a foundation for building the uniform model.

A. Static Settings

Essentially a blockchain is a peer-to-peer network among nodes. Therefore, the key factors to represent the running environment contain two parts: the nodes and the network. In this paper, a node particularly means validators in the PBFT-based blockchain consensus.

1) Nodes: To build a uniform model to facilitate fair comparison, nodes need to be configured on the same ground. That is we assume that nodes have the same hardware and network capability, and there is no difference in the relative speed i.e., $\Phi = 0$ (refer to II-C) (**Assumption 1**). For the sake of simplicity in building the model, we also assume that each node has the same amount of stake (**Assumption 2**).

The behavior of honest nodes including the leader node and other honest normal nodes follows exactly the process of PBFT as described in Fig.1. In addition, there is another type of node called the Byzantine node, which is controlled by the adversary, and can drop or delay the messages from the honest nodes, or send a wrong message to others. Because the mechanism of PBFT can tolerate up to $\frac{1}{3}$ of Byzantine nodes [13], this is simplified as the assumption that the Byzantine nodes do not take any action (**Assumption 3**).

TABLE I
A SUMMARY OF EXISTING STUDIES

WORK	OBJECTS	METRICS	METHOD	GAPS
[18] [24]	Ethereum, Hyperledger Fabric	throughput, latency	simulation	specific platforms, simple analysis, no innovation
[23]	Ethereum, Hyperledger	stability (latency), throughput	simulation	specific platforms, simple analysis, limited practical use
[31]	Ethereum, Hyperledger Fabric, Parity-PoW, CITA	memory cost, CPU cost, etc	simulation	specific platforms, too many metrics
[15]	Ethereum, Parity, Hyperledger Fabric	throughput, latency, scalability, fault-tolerance	simulation	specific platforms

2) Network: We assume a semi-synchronous network meaning that messages will be received within an upper bound and will not be delayed or dropped (**Assumption 4**). How to determine a suitable bound will be discussed later. This assumption is reasonable and practical. Although PBFT is running in an asynchronous network, meaning that there are messages delayed or dropped due to network errors, they are at a low rate for example using messages delay-or-drop tolerance technologies [19].

As mentioned earlier, in a gossip protocol, sending and receiving messages are in rounds at a given frequency. In this paper, we define the interval of this frequency as a *hop*. We assume all the messages will be sent at the beginning of a hop and receivers receive the message at the end of the same hop with a probability, and the message processing (verification, signature, and peer selecting) takes instant time and can be ignored (**Assumption 5**).

We also assume a fully connected network among validators (**Assumption 6**). Each node sends approvals and commitments to all other nodes during the PBFT process, using the gossip protocol in which the node selects the receivers from its peer list. Unlike non-stake nodes which only receive deterministic blocks and add them to the local chains, the validator nodes only take up a small proportion (e.g. around 2,000 nodes in Solana, and much fewer in many other blockchains), so assuming a network of full connection among them is justifiable.

B. Process Simplification

As illustrated in Section II, nodes in PBFT act in an asynchronous way, complicating the efficiency calculation. Therefore, we simplified the original PBFT process into a three-stage process each of which has a clear termination.

a) Challenges of Calculating with the Original Process: In the original process, nodes may be in different stages at the same time point. As shown in the original PBFT process (Fig. 1), a message takes various time periods to be received, so the same message is received at different time points by different nodes. This difference is enlarged as the propagation of the message (i.e., the receivers forward the message) in the peer-to-peer network setting. As a result, nodes are in different states, making it unavoidable to consider the state of each node when calculating the message cost.

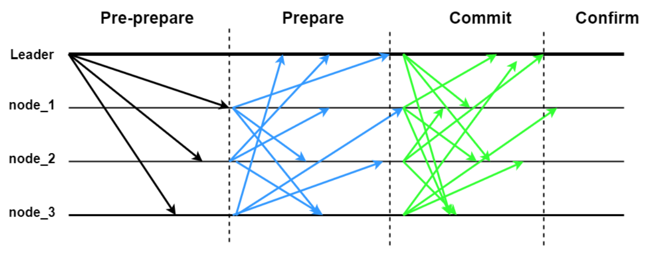


Fig. 3. Separated PBFT process

To simplify this process, we divide the original PBFT process into three separate stages where all the nodes are in the same stage at the same time as shown in Fig. 3.

1) The first stage - Pre-prepare. This stage is the same as the original description in Fig. 1 except for the condition to the next stage. An honest node in the original process only receives the created block and then enters its Prepare stage. In this model, we require each node to wait until most nodes receive the block.
2) The second stage - Prepare. This stage starts when most nodes receive the block. In addition to gossiping the verified block with its signature as approval, in this stage, every node accumulates the received non-redundant approvals from different nodes. When a node receives a supermajority of approvals, it waits instead of directly entering the next stage. Only when the supermajority of nodes receives the supermajority of approvals, all nodes enter stage three.
3) The third stage - Commit. Stage three is similar to stage two, except that the messages are commitments instead of approvals. When the termination condition is satisfied i.e., the supermajority of nodes receive the supermajority of commitments, the block is confirmed.

V. The Uniform Model

A. Configurations of the Model

Based on the above abstraction, key factors that may impact the efficiency, referred to as *variables*, are summarized in Table II. By setting up the given variables, one can get an instance of the above-abstracted model. The model has n nodes, consisting of 1 leader nodes, $f * n$ Byzantine nodes, and the remaining $(1 - f) * n - 1$ normal honest nodes. The number of Byzantine nodes is limited to $\frac{n}{3}$. The nodes behave

TABLE II
VARIABLES IN OUR MODEL

VARIABLE	DEFINITION
n	number of nodes
f	ratio of Byzantine nodes, $< \frac{1}{3}$
gossip type	SIR or SI
fanout	number of selected peers per gossip
gossip frequency	The frequency of sending messages per second

as described earlier according to their types. When a node sends messages, it follows the given gossip type (SIR or SI)—randomly selects $fanout$ number of other nodes to send at the beginning of a hop. The interval of a hop can be derived from the given gossip frequency. The entire model has three stages and each stage has its own termination condition.

Based on this model, efficiency is formalized as two properties: network load and finality. The network load measures the number of messages transmitted among honest nodes (the leader node and the normal honest nodes) to reach consensus in one round. Note that since it is assumed that the Byzantine ratio is bounded by $\frac{1}{3}$, a block can be always confirmed. However, Byzantine nodes may send fake or redundant messages that cannot be approved. This influences the total number of messages transmitted over the network, meaning that the total messages number transmitted over the network is not a good metrics to measure the efficiency of the algorithm. Therefore, we only count the messages transmitted among honest nodes. The finality measures the number of hops needed to confirm a block in a round. Recall that in related works, efficiency evaluation is based on simulations i.e., installing blockchain systems locally, then running and collecting data, while our model is theoretical and independent of hardware, which means the simulated results with absolute time do not work in our model. Therefore, we use the number of hops to represent the time, as time can be inferred by the number of gossip hops if the frequency of gossip is given.

To calculate the number of messages and hops needed to confirm a block, we first build a Bayesian network for PBFT (V-B). The messages needed to reach the state in the Bayesian network are then calculated (V-C). Particularly, we define the states of finality in the Bayesian network as probabilistic conditions for each stage. Finally, we design the respective calculation process of network load and finality during the calculation (V-D).

B. Building Bayesian Network

We use Fig. 4 to illustrate the Bayesian network of stage one, and the other two stages are a repeat of this stage (see Section V-C). In the Bayesian network, we define a state as the number of honest nodes that have received a message at hop i, named as the accumulative weight of the message at the hop i denoted as ω_i. As the number of message receivers does not decrease, the accumulated weight increases monotonically. This process is a monotonic increase process. Thus the probability of the state transaction from state ω_i to

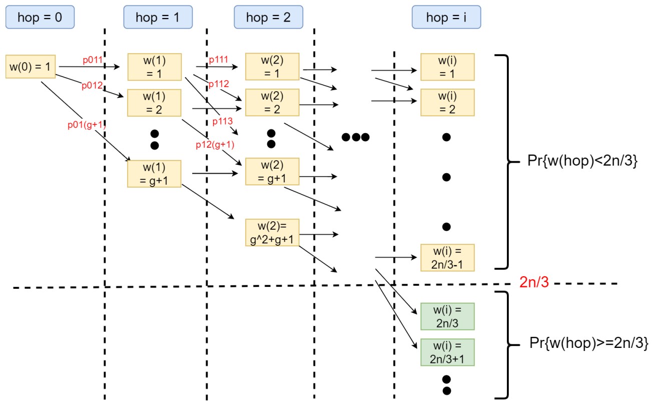


Fig. 4. calculation of Bayesian Network

TABLE III
NEWLY INTRODUCED VARIABLES

VARIABLE	DEFINITION
p_{ijk}	probability of Pr($\omega_{i+1} = k \mid \omega_i = j$) in combination calculation
P_i	probability of a message being received during i hops
I[]	an array showing the number of nodes that should receive new messages I[i] shows the received number in hop i
threshold	trigger to jump to the next stage or finalize

ω_{i+1}, denoted as $Pr(\omega_{i+1} \mid \omega_i)$ can only be either of the following two cases:

$$\begin{cases} \omega_{i+1} > \omega_i, & \omega_{i+1} - \omega_i \ \textit{new nodes receive block} \\ \omega_{i+1} = \omega_i, & \textit{no new nodes receive block.} \end{cases} \tag{2}$$

The formula means that there are ω_i nodes that have received the message of the block at the i-th hop. The message, due to the randomness of gossip, is possible to be sent to nodes that have already received it, which results in redundancy. At the $(i+1)$-th hop, the weight ω_{i+1} increases if there are other non-received nodes receive this message, which leads to $\omega_{i+1} > \omega_i$, or ω_{i+1} keeps unchanged if no other non-received nodes receive this message, which leads to $\omega_{i+1} = \omega_i$. This is possible because the PBFT-based blockchains such as Hyperledger Fabric [9], Dfinity [17], Solana [29], and Hedera [10] adopt the gossip protocol where a node may send redundant messages to the nodes who have already received it, which does not lead to an increase of ω.

When $\omega_{i+1} > \omega_i$, the calculation method of `combination`(C_n^m) can be used to calculate the subsequent probabilities of the specific increases. Based on this, the states in the next hop are only dependent on the states in the current hop, it satisfies the Bayesian network of its conditional probability distribution, and then the confirmation process of a block of the PBFT consensus – to accumulate at least $\frac{2}{3}$ of nodes who have received and verified the block – can be formulated as a Bayesian network (layered by hops).

To facilitate the understanding and calculation, we introduce some new variables shown in Table III. The distribution P_i is a normal distribution with its mean and deviation (will be discussed in Subsection VI-D). The array I[] is an accumulation array. If there is a non-zero probability that some new nodes receive the message at the hop i, the value I[i] is added by the probability times the base number, when the hop i is passed, the accumulated value of I[i] is the total change of ω.

As the stages are both probabilistic processes (Section IV), the *threshold* is set as a role that we regard an event as definitely happening when its probability of happening is more than the threshold (**Assumption 7**). Here we set the threshold as 99%.

C. Calculations for Each Stage

1) Pre-prepare stage: In this stage, the leader node creates a block, packs it in a message, and sends the message to other nodes. So the initial state is $\omega_0 = 1$, as only the leader node has the message. The condition of switching to stage 2 is set that only when 99% of honest nodes receive the block, the algorithm enters the next stage.

At the beginning of the first hop, the leader sends a message to g nodes—the number of fanout. Therefore, ω_1 ranges from $\omega_1 = 1$ to $\omega_1 = 1 + g$, capturing that some of the receivers do not receive the message sent by the leader nodes in one hop. We assume that the distribution of the probability of $Pr(\omega_1 = 1)$ to $Pr(\omega_1 = 1 + g)$ follows a given probability distribution. Based on the distribution, the probability of each state can be calculated. For example, in Fig. 4, the conditional probability of $Pr(\omega_1 = 3)$ is calculated as follows:

$$\begin{aligned} Pr(\omega_1 = 3) = & Pr(\omega_0 = 1) * p_{013} * P_0 + \\ & Pr(\omega_1 = 3) * p_{133} * (1 - P_0) \end{aligned} \tag{3}$$

This process is repeated for each state from ω_0 to ω_1,..., ω_i, and the Pr($\omega_i = m$) can be generalized as:

$$Pr(\omega_y = k) = \sum_{j=1}^{k} \sum_{i=0}^{y} Pr(\omega_i = j) * p_{ijk} * \sum_{a=0}^{y-i-1} P_a \tag{4}$$

The trigger to switch to the next stage based on a threshold *th* of the Pre-prepare stage is formulated as:

$$Pr(\omega_i = 0.99n) >= th$$

2) Prepare stage: In this stage, the condition for a node to be ready to promote to the next stage is that it receives at least $\frac{2}{3}n$ of approvals from other nodes; and the condition for the system to promote to the next stage is $\frac{2}{3}n$ of honest nodes are ready for the next stage. This can be formulated, based on the Bayesian network as follows:

- For a single node to be prepared, the condition is Pr(ω_i) $>= \frac{2}{3}n$ at a hop i, which is the sum of all the probabilities of states where $\omega_i >= \frac{2}{3}n$, formulated as follows:

$$Pr(\omega_i >= \frac{2}{3}n) = \sum_{k=\frac{2}{3}n}^{n} Pr(\omega_i = k) \tag{5}$$

- For the system, due to the assumption that all the nodes are in the same ground as mentioned in IV-A and independent, the condition to promote to the next stage is formulated as:

$$Pr(\omega_i >= \frac{2}{3}n)^{\frac{2}{3}n} >= th \tag{6}$$

It means that the probability that over $\frac{2}{3}n$ of honest nodes are ready for the next stage needs to be larger than the given threshold.

3) Commit stage: The Commit stage works the same as the Prepare stage except for the message content (commitments instead of approvals), thus it holds the same termination condition, and the same calculation process.

D. Efficiency Analysis

In this subsection, calculations of them are analyzed, in order to acquire data on the efficiencies of blockchains, as well as explore the impact of variables.

1) Finality: The finality is represented by the number of hops and can be translated to real-time by the accordingly gossip setting. We record the number of hops when the termination condition for each stage is satisfied and denote them as h_1, h_2, and h_3 respectively. The total number of hops is:

$$H = h_1 + h_2 + h_3 \tag{7}$$

Given the gossip frequency, the real-world time cost in seconds can be derived using $H/frequency$.

2) Network Load: When an honest node receives an updated message, it transforms its state to I, and runs the gossip once to transfer that message, and then its state holds (in the SI model), or is transformed to R (in the SIR model) and this node does not transfer the complete same message anymore.

During the process of calculations of stages, the variable I[] is always updated during per hop:

$$I[y] = \sum_{i=0}^{y} I[i] * g * p_{iI[i]I[y-1]} * P_{y-i} \tag{8}$$

According to I[], the message cost C_i in the hop i with the *fanout* g can be equated as:

$$C_i = I[i] * g \tag{9}$$

and then we set W_1 as the accumulations of message cost of every hop of n_1, so it can be equated as:

$$W_1 = \sum_{i=1}^{H} C_i \tag{10}$$

where t is the number of the last hop. Finally, because all the nodes are assumed to be in the same conditions, the total message cost W can be equated as:

$$W = \sum_{m=1}^{t} W_m \tag{11}$$

Normally, the less message complexity of a blockchain reflects the lower network overload demands.

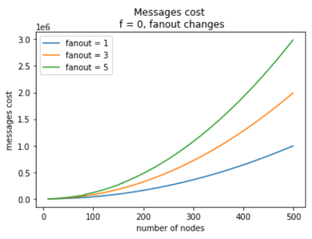


Fig. 5. message cost with various fanouts

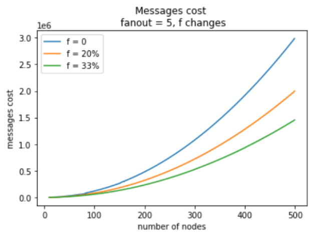


Fig. 6. message cost with various f

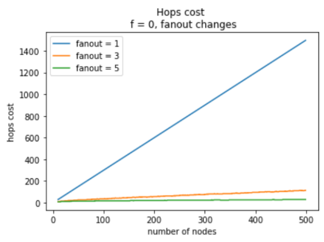


Fig. 7. hop cost with various fanout

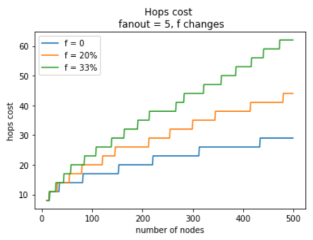


Fig. 8. hop costs with various f

VI. Analysis and Discussions

Based on the above model, the influence of variables on efficiency including network load and finality can be analyzed. The results in subsections VI-A and VI-B are based on the SIR gossip model as the majority of PBFT-based blockchains adopt the SIR model, and in subsection VI-C we discuss the comparison of results between the SI model and the SIR model.

In the analysis, the variables mentioned in the last section are inputs of our model. Variables are used to represent different blockchains, and numerical variables (n, *fanout*, f) can be adjusted are the analysis target to reveal their influence on efficiency in a given blockchain.

A. Network Load

From the shapes of the curves in Fig. 5 and Fig. 6, we can infer that the message complexity for reaching a consensus of a block is $O(n^2)$, which confirms the message complexity of the original PBFT. Since there is a limited number of transactions inside a block, assume a block has m (a constant upper bound) transactions, the average message complexity for a transaction to reach consensus is $\frac{O(n^2)}{m}$ which is still $O(n^2)$. (Finding 1)

Fig.5 shows the message cost when the *fanout*s are 1, 3, and 5 respectively. From the figure, we can see that there is a positive correlation between the *fanout* and the message cost, the higher *fanout* will lead to the higher message cost, and vice versa. (Finding 2)

From Fig. 6, we can see that there exists a negative correlation between the Byzantine nodes ratio and the message cost when Byzantine nodes ratio is less than $\frac{1}{3}$, which means the higher Byzantine nodes ratio will cause the lower message cost, based on our mentioned setting that we only count the messages sent by the honest nodes but not the Byzantine nodes. (Finding 3)

B. Finality

From Fig.7, it is easy to see that the cost with higher *fanout* are much lower relatively, so there is a positive correlation between the *fanout* and the hop cost (Finding 4). We can also see that the correlation between the number of nodes and the hop cost is linear when the *fanout* is equal to 1, however, they are not all linear. For example, when *fanout* =5, as shown in Fig.9, the correlation between the number of nodes and hops is sub-linear. One may observe that the hop cost curve is a

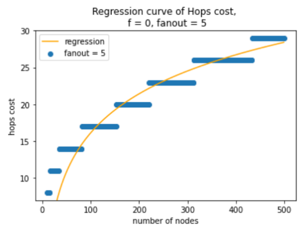


Fig. 9. Detailed hop cost with fanout = 5 and its regression

non-continuous function. This is because the number of hops can only be integers (i.e., discrete values).

From Fig.8 we observe that there exists a positive correlation between the Byzantine nodes ratio and the hop cost when the Byzantine nodes ratio is less than $\frac{1}{3}$, which means the higher Byzantine nodes ratio results in a higher hop cost, in another word, prolong the finality, and vice versa. (Finding 5)

Therefore, the finality is $O(n)$ when the *fanout* is equal to 1, or $O(log_n)$ when the *fanout* is higher than 1. (Finding 6)

Discussion: Because we separate the 3 interleaving stages from the original PBFT process, the duration of our model should be an upper bound of the original. If t_1, t_2, t_3 respectively represent the time cost of each stage, we set T as the original time cost, and T' as our model's time cost, there will be $T' >= T$, which can be shown by Fig.10:

And we can select either of the 3 stages, such as stage 2, as the lower bound of the original process because the whole duration can not be shorter than either of the sub-duration. Then because T' = t_1 + t_2 + t_3, at least one of [t_1, t_2, t_3] has the same hops complexity as T', and T $>=$ the max[t_1, t_2, t_3], so T' has the same hops complexity as well as the time complexity to T, and at most 3 times as long as T.

C. SI Gossip Model

The results and summaries above are based on the SIR gossip model, in this section, we discuss the differences in the SI model. As mentioned in the Background Section, the SI model differs from the SIR model in that all the Infected nodes do not transform to the Removed, and always keep gossiping. Therefore, with the same variables, because there are more nodes in gossiping, intuitively, the SI model can reach the finality faster than the SIR model, which is shown in Fig. 11 from the analysis, as well as the higher message cost, which can be referred to in Fig. 12.

We can see that with the same f and *fanout* in Fig. 12, the complexity of the message cost of the SI model is also $O(n^2)$

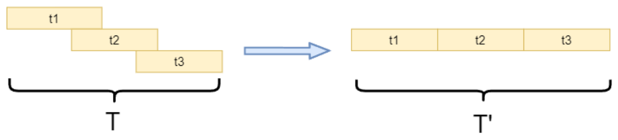


Fig. 10. The extent of total time cost

and a little higher than the SIR model at a constant level [1] (Finding 7).

We also find in the SI model that the impacts of *f* and *fanout* nearly follow the same trend as in the SIR model, which is that higher *f* brings higher hop cost and lower message cost, and the higher *fanout* brings the lower hop cost. Refer to Fig. 13, lower *f* leads to a lower hop cost, but a smaller range compared with the SIR model shown in Fig. 8. (Finding 8)

The only difference is that in the SI model the change of *fanout* impacts a little on the message cost. As for the reason, the higher *fanout* brings the higher message cost per hop, but also the lower hop cost; and a lower *fanout* has a lower hoply message cost, but higher hops, so they offset each other. (Finding 9)

D. Section Summary

During the experiments, we find that it is impossible to optimize both the time complexity and the message complexity simultaneously, and a trade-off inside the efficiency can be applied depending on the demands of the blockchain (Finding 10). The *fanout* acts as the key role to impact the trade-off between the time complexity and the message complexity, the higher *fanout* leads to a lower finality, but also a higher network load (except in the SI model), and vice versa. This can be taken into consideration when designing the consensus of a blockchain. While the Byzantine nodes ratio can also impact the network load and the finality, it is not the inner attribute of the blockchain itself.

VII. Case Study

In this section, we selected 3 popular blockchains as case studies including Solana, Dfinity, and Hedera. Solana is used to validating the accuracy of our model; then our model is used to testify to the authenticity of Dfiniy's claims and infer unknown information about Hedera, which is summarized in Fig. 16.

A. Settings and Assumptions

a) Stake distribution: Our model assumes that each node has the same stake (Assumption 8). In reality, none of the 3 blockchains satisfy the assumption. However, the stake distribution in the 3 blockchains is relatively balanced among a set of nodes. The stake distribution can be measured by the GINI coefficient. Solana's GINI coefficient is 0.622, and Dfinity is 0.678. This is a relatively balanced distribution, compared to Bitcoin with a GINI coefficient of 0.968 (using data in 2018). As for Hedera, according to its whitepaper [10], every node currently has the same stake, the same as our assumption.

[1]The sudden jump is due to error accumulation.

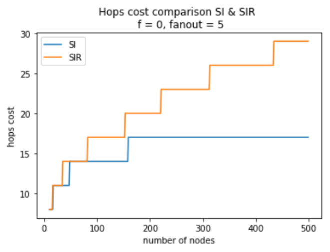


Fig. 11. hop cost comparison between SI and SIR

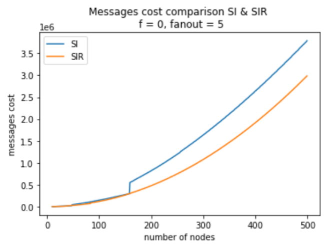


Fig. 12. message cost comparison between SI and SIR

b) Byzantine ratio: Solana is a public blockchain platform, which means anyone including the malicious node can join as a validator. However, it is impossible to figure out the Byzantine ratio. Therefore, we assume 0 Byzantine nodes in our analysis, since being malicious is discouraged by their incentive and no malicious validator has been reported. As for Dfinity and Hedera, since they are permissioned blockchain platforms—nodes are authenticated before joining, and the chance for a node to be malicious is even smaller. Thus it is safe to assume that *f*=0.

c) Gossip frequency: The variable gossip frequency is 10 per second, as stated [29], which is the default setting of gossip protocols [20]. There is no such statement for Dfinity and Hedera, we adopt the default setting as well.

Once the gossip frequency is set, the hop can be derived. Recall that hop is defined as a round of gossip sending and receiving whose value is the same as the interval of the gossip frequency, which will be 100 milliseconds by default. This interval is long enough to assume a message is always delivered within a hop, i.e., 100 milliseconds is much higher than the imperial upper bound Δ. This is derived from the gossip protocol experiments and simulations in [12], which set up a set of nodes in 13 states in the US through AWS and collected data to measure the latency of message delivery among nodes under a voting-based consensus. We use the data of the experiment with the most number of nodes—105 nodes in total. As described in the experiment, every node has 6 ($logn$) neighbors and sends messages to all the neighbors during a gossip. To send to and receive the responses from all the other nodes, we can derive that the number of hops is at least 6. According to the experiment results, the average total time cost of a gossip process is 280ms with the standard deviation being 50ms. We can derive that the time per hop is around 47ms ($280ms/6$) and the standard deviation is around 8ms($50ms * \sqrt{n/(n-1)}/6$). Assume the probability of the message transition time follows a normal distribution, $100ms > 47ms + 6 * \sigma$, the probability that a gossip message is received within 100ms is more than the *threshlod* 99.99%, so we assume that it definitely happens. Therefore, a message must be delivered within 1 hop, which means P_0 is 100%.

B. Solana

Solana's official website [8] [6] and its open-source code [7] provide the following data: the *fanout* is 6, number of nodes is around 2016, the finality in 6 hours is $2.0s - 6s$ with the

TABLE IV
VARIABLES OF BLOCKCHAIN PLATFORMS, SAMPLING TIME: 17/10/2022

Platform	n	fanout	gossip type	gossip frequency
Solana	2016	6	SIR	10/s
Dfinity	826	20	SIR	10/s
Hedera	unknown	1	SI	10/s

majority being $2s - 3.5s$, the gossip type is SIR, as shown in Table.IV. Therefore, Solana is a perfect case study to validate the accuracy of our model.

When we input those variables into our model, the results show that the estimated finality hops are 35, and the estimated finality time is 3.5s with the given gossip frequency of 10 times per second, which matches the real-world data. The network load is not stated in Solana. According to our analysis, the message cost per block in Solana is around 65 million, shown in Fig. 16.

Discuss: In Solana, a node keeps the record of the message sender to detect redundant messages, while in our model we randomly select receivers and redundant messages may exist. Therefore, the finality, as well as the network load, calculated from our model may be slightly higher than the real-world data. This can be a reason that our estimated finality—3.5s is at the top margin of the majority finality range $2s - 3.5s$.

C. *Dfinity*

Dfinity claims its finality can reach an average of 1 second on its main net, while there is not any official statistic to show it. In this subsection, we utilized our model to testify the claim.

Dfinity's *fanout* on the main net is 20 (named as advert in) [1], and the total number of nodes is around 826 [3], its claimed finality is between 1s to 2s [17] even the 1-second finality [2], its gossip type is SIR. From the results of our model, the finality is 11 hops, i.e., 1.1s, which confirms its claim. From our model, Dfinity's network load is around 12.5 million.

D. *Hedera*

Differing from the above two blockchains, Hedera uses SI gossip.

Hedera's *fanout* is 1 [10], finality is around 4.5 seconds [4]. However, there is no information on the number of validators. By applying our model, the number of validators can be inferred—between 25 and 35 as shown in Fig. 15 (we pick the median which is 27). The network load is around 1.2 thousand.

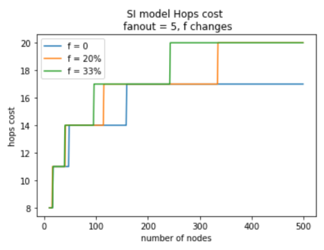


Fig. 13. hop cost of SI model

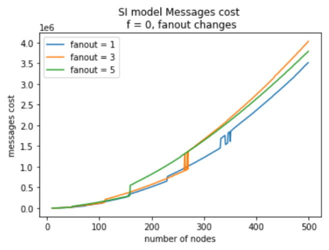


Fig. 14. message cost of SI model

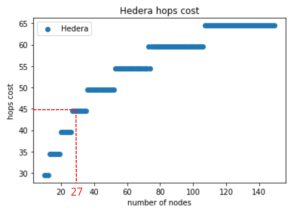
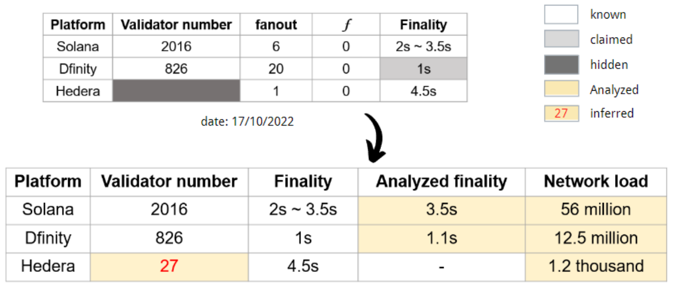


Fig. 15. Hedera: validator number deduction



Platform	Validator number	fanout	f	Finality
Solana	2016	6	0	2s ~ 3.5s
Dfinity	826	20	0	1s
Hedera	■	1	0	4.5s

date: 17/10/2022

☐	known
▨	claimed
■	hidden
🟨	Analyzed
27	inferred

Platform	Validator number	Finality	Analyzed finality	Network load
Solana	2016	2s ~ 3.5s	3.5s	56 million
Dfinity	826	1s	1.1s	12.5 million
Hedera	27	4.5s	-	1.2 thousand

Fig. 16. Cases summary

Discuss: Note that the size of messages of Hedera (SI gossip) is much larger than Solana and Dfinity (SIR gossip) because they contain historical data when synchronizing. As a result, the computing load of nodes is higher in Hedera. On the other hand, the network load of Hedera doesn't increase when the number of blocks increases, but Solana and Dfinity definitely do, and the average transaction message complexity on the Hedera is actually $\frac{O(n^2)}{trans}$ instead of $O(n^2)$ in the general PBFT blockchains, where 'trans' is the number of currently unconfirmed transactions in that gossip and is not a constant level.

E. *Potential Applications*

The outcomes of our model can be potentially useful in both academics and industry. We list a few potential applications that can benefit the mutual trust between blockchain platforms and clients, contribute to a better design of blockchain systems, and improve safety by monitoring the change of malicious nodes.

1) Infer Hardware Requirements: Given the results from our model, the minimum hardware requirements for anode can be inferred, using the average amount of messages per second that need to be dealt with by a single node calculated from the analyzed network load. Designers can use this as a guide to suggest the least level of hardware to handle the network load. On the other hand, if designers have the information on hardware, the model can be used to estimate if their efficiency requirements can be achievable.

2) Verify the Authenticity of Platforms: With the values of the key factors of a certain blockchain, a user can input these values into our model and verify the provided information i.e., whether the claimed efficiency matches the analysis results (same as in the case study Solana).

3) Supporting configuration of blockchain: The identified key factors and their impacts on efficiency can support blockchain designers in decision-making. For example, the designers can shorten finality by increasing the setting of the fanout.

4) Byzantine Nodes Ratio: The correlation between the Byzantine node rate and efficiencies derived from our model can be used to get hints of the Byzantine node ratio. If a stable decline in efficiencies is observed in a blockchain platform, while the value of all other values remains, it is a hint that the Byzantine nodes rate is increasing.

5) Reveal Hidden Values of Key Factors: It is common that some blockchain platforms do not reveal the values of some key factors, which hinders blockchain analysis or comparison. In the case that the value of one variable is missing, our model can be used to reveal some hidden information from the values of the other variables. In the case of multiple values are missing, our model can be used to reveal the correlation among the multiple hidden variables.

VIII. Conclusion and Future Work

In this paper, we built an innovative model to analyze the process of achieving the consensus of PBFT-based blockchains, and based on the experiments we find some specific impacts of variables to the efficiencies, those impacts can be considered when designing a new blockchain. We also find that there exists a trade-off between the network load and the finality, and it can be adjusted by adjusting the value of the *fanout*. Finally, we picked up some popular blockchain platforms as the case studies, extracted their variables as the inputs to our model, and match our analysis results and the real data, to prove the accuracy of our model, and based on the pass-verified model we can infer some hidden information of a blockchain in reality by inputting other matched attributes into our model, such as the number of validators.

In future works, we plan to take more PBFT-based blockchains into consideration and provide a fair comparison among platforms. In addition, we will explore the possibility to apply the type of theoretical analysis method in this paper to other non-PBFT consensus blockchains.

References

[1] Dfinityfanout. https://github.com/dfinity/ic/blob/cce49b4c100d0ecc205664415c30c5441a760d15/rs/types/types/src/p2p.rs. Accessed: 2022-10-17.

[2] Dfinityfinality. https://medium.com/dfinity/the-internet-computers-transaction-speed-and-finality-outpace-other-l1-blockchains-8e7d25e4b2ef. Accessed: 2022-10-17.

[3] Dfinitynodes. https://dashboard.internetcomputer.org/nodes. Accessed: 2022-10-17.

[4] Hederafinality. https://app.metrika.co/dashboard/hedera/network-overview?tr=1d. Accessed: 2022-10-17.

[5] Pwc: blockchainmarket. https://www.pwc.com/gx/en/industries/technology/publications/blockchain-report-transform-business-economy.html. Accessed: 2022-12-16.

[6] Solanabeach. https://solscan.io. Accessed: 2022-10-17.

[7] Solanafanout. https://github.com/solana-labs/solana/blob/v1.14/gossip/src/crds_gossip_push.rs. Accessed: 2022-10-17.

[8] Solananodes. https://solana.com/zh/validators. Accessed: 2022-10-17.

[9] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[10] Leemon Baird, Mance Harmon, and Paul Madsen. Hedera: A public hashgraph network & governing council. *White Paper*, 1, 2019.

[11] Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform. *Whitepaper*, 2013.

[12] Daniel Cason, Nenad Milosevic, Zarko Milosevic, and Fernando Pedone. Gossip consensus. In *Proceedings of the 22nd International Middleware Conference*, pages 198–209, 2021.

[13] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[14] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[15] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. Blockbench: A framework for analyzing private blockchains. In *Proceedings of the 2017 ACM international conference on management of data*, pages 1085–1100, 2017.

[16] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[17] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

[18] Yue Hao, Yi Li, Xinghua Dong, Li Fang, and Ping Chen. Performance analysis of consensus algorithm in private blockchain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 280–285. IEEE, 2018.

[19] Amir Krifa, Chadi Barakat, and Thrasyvoulos Spyropoulos. Optimal buffer management policies for delay tolerant networks. In *2008 5th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks*, pages 260–268. IEEE, 2008.

[20] Marta Kwiatkowska, Gethin Norman, and David Parker. Analysis of a gossip protocol in prism. *ACM SIGMETRICS Performance Evaluation Review*, 36(3):17–22, 2008.

[21] Du Mingxiao, Ma Xiaofeng, Zhang Zhe, Wang Xiangwei, and Chen Qijun. A review on consensus algorithm of blockchain. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2567–2572. IEEE, 2017.

[22] Alberto Montresor. Gossip and epidemic protocols. *Wiley encyclopedia of electrical and electronics engineering*, 1, 2017.

[23] Santosh Pandey, Gopal Ojha, Bikesh Shrestha, and Rohit Kumar. Blocksim: A practical simulation tool for optimal network design, stability and planning. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 133–137. IEEE, 2019.

[24] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. Performance analysis of private blockchain platforms in varying workloads. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. IEEE, 2017.

[25] Mahendra Kumar Shrivas and Dr Thomas Yeboah. The disruptive blockchain: Types, platforms and applications. In *Fifth Texila World Conference for Scholars (TWCS) on Transformation: The Creative Potential of Interdisciplinary*, 2018.

[26] Todd Andrew Stephenson. An introduction to bayesian network theory and usage. Technical report, Idiap, 2000.

[27] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*, 21:2327–4662, 2016.

[28] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.

[29] Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*, 2018.

[30] Manuel Zander, Tom Waite, and Dominik Harz. Dagsim: Simulation of dag-based distributed ledger protocols. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):118–121, 2019.

[31] Peilin Zheng, Zibin Zheng, Xiapu Luo, Xiangping Chen, and Xuanzhe Liu. A detailed and real-time performance monitoring framework for blockchain systems. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 134–143. IEEE, 2018.